



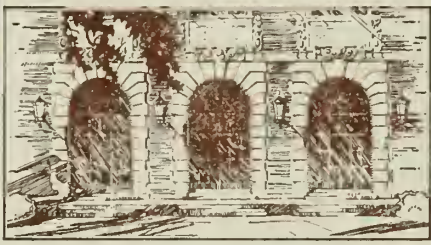
LIBRARY OF THE  
UNIVERSITY OF ILLINOIS  
AT URBANA-CHAMPAIGN

510.84

Il 6r

no. 812 - 817

cop. 2



## CENTRAL CIRCULATION BOOKSTACKS

The person charging this material is responsible for its renewal or its return to the library from which it was borrowed on or before the **Latest Date** stamped below. **You may be charged a minimum fee of \$75.00 for each lost book.**

Theft, mutilation, and underlining of books are reasons for disciplinary action and may result in dismissal from the University.

TO RENEW CALL TELEPHONE CENTER, 333-8400

UNIVERSITY OF ILLINOIS LIBRARY AT URBANA-CHAMPAIGN

MAR 11 1998

When renewing by phone, write new due date below previous due date.

L162



Digitized by the Internet Archive  
in 2013

<http://archive.org/details/numericalsolutio813link>





NUMERICAL SOLUTION OF STIFF ORDINARY DIFFERENTIAL EQUATIONS  
USING COLLOCATION METHODS

by

BRUCE DAVID LINK

*BDL*

June 1976



DEPARTMENT OF COMPUTER SCIENCE  
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

THE LIBRARY of the  
SEP 14 1976  
University of Illinois  
at Urbana-Champaign



NUMERICAL SOLUTION OF STIFF ORDINARY DIFFERENTIAL EQUATIONS  
USING COLLOCATION METHODS

by

BRUCE DAVID LINK

June 1976

DEPARTMENT OF COMPUTER SCIENCE  
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN  
URBANA, ILLINOIS 61801

This work was supported in part by the Energy Research and Development Administration under contract US ERDA E(11-1) 2383 and submitted in partial fulfillment of the requirements of the Graduate College for the degree of Doctor of Philosophy in Computer Science.



## ACKNOWLEDGEMENTS

I am very grateful to my advisor Daniel S. Watanabe for his many valuable suggestions and comments during the course of this project. He was always available and willing to help with problems. I am also grateful to C. W. Gear for many helpful discussions. My wife, Regina Link, was understanding and extremely patient during the work. In addition, she prepared Figs. 1.1-1.3 and typed the manuscript. The work was supported in part by the Energy Research and Development Agency under grant US ERDA-E(11-1) 2383 and the Department of Computer Science of the University of Illinois.

This work is dedicated to my son, Hamilton Edmund Link.



## TABLE OF CONTENTS

	Page
1. INTRODUCTION: NONLINEAR MULTISTEP METHODS. . . . .	1
1.1 Notation and Definitions . . . . .	8
2. SIMPLE COLLOCATION METHODS. . . . .	15
2.1 Order. . . . .	15
2.2 Stability Regions. . . . .	24
2.3 Implementation Considerations. . . . .	30
3. BLOCK COLLOCATION METHODS . . . . .	38
3.1 Order. . . . .	38
3.2 Stability Regions. . . . .	44
3.3 Stability and Convergence. . . . .	58
4. QUADRATURE METHODS. . . . .	72
LIST OF REFERENCES. . . . .	79
APPENDIX I: LISTING OF CODE FOR COLLOC1 FAMILY OF METHODS. . . . .	81
APPENDIX II: LISTING OF CODE TO FIND AND PLOT STABILITY REGIONS . . . . .	92
VITA. . . . .	106



NUMERICAL SOLUTION OF STIFF ORDINARY DIFFERENTIAL EQUATIONS  
USING COLLOCATION METHODS

Bruce David Link, Ph. D.  
Department of Computer Science  
University of Illinois at Urbana-Champaign, 1976

A new class of methods, collocation methods, is suitable for the integration of stiff ordinary differential equations. The order and stability regions of these methods are characterized. The methods are stable and convergent for infrequent formula changes and small stepsize changes. Block multistep methods which are stiffly stable up to order 24 and several families of multistep methods stiffly stable up to order 9 exist. A similar class of quadrature methods includes a family with the same stability regions as Enright's second derivative methods but only requires first derivatives.



## 1. INTRODUCTION: NONLINEAR MULTISTEP METHODS

Consider the ordinary differential equation initial value problem

(ODE)

$$y' = f(y, t) , \quad (1.1)$$

$$y(t_0) = y_0 ,$$

where

$$f : \mathbb{R}^{n+1} \rightarrow \mathbb{R}^n ,$$

$$y : [t_0, T] \rightarrow \mathbb{R}^n .$$

We shall assume that  $f$  satisfies a Lipschitz condition in  $y$  and is continuous, so that a unique solution of the ODE exists and is in  $C^1[t_0, T]$ . Later we shall further assume that  $f$  has bounded continuous partial derivatives with respect to the  $y$  variables and that the solution has as many continuous derivatives as is necessary.

Differential equations which arise in the description of physical problems often have widely differing time constants. We call an ODE stiff if the set of eigenvalues of the Jacobian  $\partial f / \partial y$  includes some members whose real parts are much more negative than those of other members. For example, in the equation

$$y_1' = -500.5 y_1 + 499.5 y_2 ,$$

$$y_2' = 499.5 y_1 - 500.5 y_2 ,$$

$$y_1(0) = 2 ,$$

$$y_2(0) = 0 ,$$

(1.2)

the exact solution is

$$y_1(t) = e^{-t} + e^{-1000t} ,$$

$$y_2(t) = e^{-t} - e^{-1000t} .$$

The equation can be written in the form

$$y' = P \Lambda P^{-1} y ,$$

where

$$P = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad \Lambda = \begin{pmatrix} -1 & 0 \\ 0 & -1000 \end{pmatrix} .$$

Because the eigenvalue  $-1000$  is much more negative than the eigenvalue  $-1$ , the equation is stiff.

Stiff equations often occur in the analysis and simulation of networks, in almost all chemical kinetic studies, and in control problems, especially when system states must be modeled (Bjorel et al., 1970). In the solution of a stiff equation the components corresponding to some eigenvalues become negligibly small while others are still large enough to be of interest. This would be true in the above example for  $t$  larger than 0.05.

Methods for numerically solving ODE's calculate a sequence of approximations,  $\{y_n\}$ , to the solution of equation (1.1) on a mesh of time values  $\{t_n\}$  where  $t_0 < t_1 < \dots < t_N = T$ . The stepsizes are defined to be

$$h_n = t_n - t_{n-1} ,$$

$$h = \max_n \{h_n\} .$$

Most methods operate serially and at each step use the past  $r$  points,

$$(y_{n-r}, t_{n-r}), \dots, (y_{n-1}, t_{n-1}) ,$$

to calculate the next  $s$  approximations,

$$(y_n, t_n), \dots, (y_{n+s-1}, t_{n+s-1}) .$$

The specific values of  $r$  and  $s$  and the stepsizes  $h_n$  are often varied in order to satisfy an error tolerance.

In solving a stiff ODE numerically, one would like to choose step-sizes according to the dominant components of the solution. Most methods not specifically designed for stiff ODE's must restrict the stepsize for stability reasons, with the stepsize depending on the eigenvalue with the greatest magnitude. For example, Euler's formula defines  $y_n$  in terms of  $y_{n-1}$  as

$$y_n = y_{n-1} + hf(y_{n-1}, t_{n-1}) .$$

If we apply Euler's formula to the ODE (1.2) the solution is

$$\begin{aligned} y_n &= (I + hPAP^{-1})^n y_0 = P(I + h\Lambda)^n P^{-1} y_0 \\ &= P \begin{bmatrix} (1-h)^n & 0 \\ 0 & (1 - 1000h)^n \end{bmatrix} P^{-1} y_0 . \end{aligned}$$

We see that for the numerical solution to converge to zero as  $t \rightarrow \infty$  (as the exact solution does), we must restrict  $h \leq 0.001$ , due to the presence of the eigenvalue  $-1000$ . The stepsize must be small even after the component due to this eigenvalue is negligible, because roundoff errors will always reintroduce this component into the numerical solution. Thus the stability properties of Euler's method force the stepsize to be 1000 times smaller than necessary to accurately follow the component of the solution actually present for  $t > 0.03$ .

This thesis presents some new classes of formulas suitable for stiff ODE's. To discuss these formulas and compare them, we need a few definitions. We say intuitively that a formula is of order  $k$  if the error

$$e_n = y_n - y(t_n)$$

satisfies

$$e_n = O(h^k) \text{ as } h \rightarrow 0 ,$$

for sufficiently smooth  $f$ . This definition mixes the concepts of order and stability. To avoid this, the definition is usually stated in the one step form: a formula is of order  $k$  if, given past values  $y_{n-i} = y(t_{n-i})$ , the future values calculated for the next step satisfy

$$c_{n+i} = O(h_{n+i}^{k+1}), \quad i = 0, \dots, s-1.$$

A method is stable for a stepsize selection scheme and (possibly) a formula selection scheme, both depending on  $h$ , if there exists an  $h_0 > 0$  for each differential equation such that a change in the starting values by a fixed amount produces a bounded change in the numerical solution for all  $0 < h \leq h_0$ . This definition, like that for order, is concerned only with the method's behavior for sufficiently small stepsize. In practice we are concerned with larger  $h$ , and would often like to make  $h$  as large as possible. A useful concept is that of the region of absolute stability, defined as the set of  $h$  and  $\lambda$  for which a variation in a single value,  $y_n$ , will produce a change in subsequent values which does not increase from step to step when the formula is applied to the test equation

$$y' = \lambda y$$

with fixed stepsize  $h$ . In the formulas considered in this study, the regions of absolute stability will have the form

$$\{(h, \lambda) \mid h\lambda \in B \subset \mathbb{C}\}.$$

One usually refers to  $B$  as the region of absolute stability, for ease in plotting stability regions.

A formula is called  $A(\alpha)$ -stable for  $\alpha \in (0, \pi/2]$  if the numerical approximations for  $y' = \lambda y$  converge to 0 as  $n \rightarrow \infty$ , with constant stepsize  $h$  and all nonzero  $\lambda$  satisfying  $|\text{Arg}(-\lambda)| < \alpha$ . Thus the region of absolute stability contains the wedge cross-hatched in Fig. 1.1.

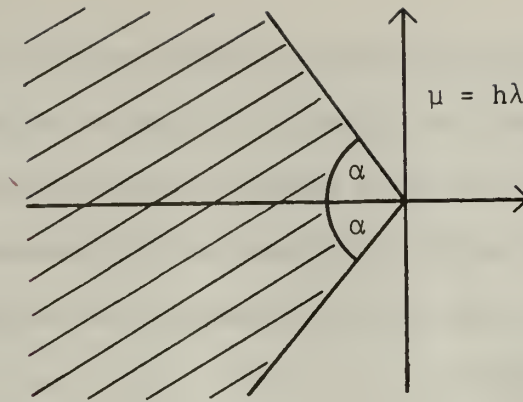


Fig. 1.1. Region of absolute stability for an  $A(\alpha)$ -stable formula

A formula is called A-stable if it is  $A(\pi/2)$  stable and strongly A-stable if it is A-stable and

$$\lim_{h\lambda \rightarrow -\infty} \limsup_{n \rightarrow \infty} y_n / y_{n-1} < 1$$

where  $\{y_n\}$  is the solution of  $y' = \lambda y$  for stepsize  $h$ .

Finally, a formula is called stiffly stable if there is a  $D$  such that the formula is absolutely stable in the region  $\text{Re}(h\lambda) < D$ , accurate around the origin, and  $A(\alpha)$ -stable for some positive  $\alpha$ . The region of absolute stability will then contain a region such as that shown in Fig. 1.2.

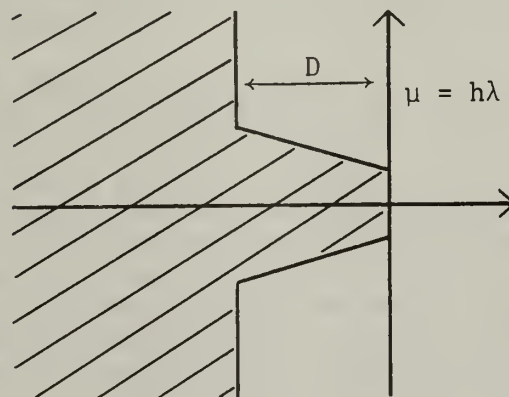


Fig. 1.2. Region of absolute stability for a stiffly stable formula

Until recently, formulas suitable to stiff ODE's were limited to very low order. Before Gear's work (1969), most of these formulas had

order at most two (Bjorel, 1970). Methods which were stiffly stable with higher order were one step methods (e.g., implicit Runge-Kutta), which were less efficient since they could not use past values to increase the order. Gear recognized that the backward differentiation formulas would be useful for stiff ODE's. Gear's formulas are a special case of the linear multistep formula

$$\alpha_{-r,0}y_{n-r} + \dots + \alpha_{0,0}y_n + \alpha_{-r,1}h_n y'_{n-r} + \dots + \alpha_{0,1}h_n y'_n = 0.$$

They are obtained by setting  $\alpha_{-r,1} = \dots = \alpha_{-1,1} = 0$ ,  $\alpha_{0,1} = -1$ , and choosing the  $\alpha_{i,0}$  for maximal order. These formulas were discussed by Henrici (1962), who dismissed them as not being useful. The formulas are called backward differentiation formulas because the  $\alpha_{i,0}$  come from differentiating the polynomial interpolating the values  $y_{n-r}, \dots, y_n$ . The backward differentiation formulas are stiffly stable for order up to 6. The most popular codes for solving stiff equations are based on these formulas. The formula of order 6 has an  $\alpha$  of only  $17.9^\circ$ , however, and many people use only the first five. The restriction to low order has stimulated a search for better formulas that give stiff stability for higher order.

It has been an open question whether there exist high order stiffly stable formulas which are practical. Cryer's k-step multistep formulas (Cryer, 1973) have been shown by Jeltsch (1976) to be stiffly stable for arbitrarily high order. These methods are not practical, however, since the regions of instability become so small that the stepsize must be made very small to accurately follow the dominant part of the solution. The  $\alpha$ 's given by Jeltsch (1976) are also inferior to the  $\alpha$ 's of the BLOCK3 family given in Chapter 3 of this paper.

Jain and Srivastava (1970) found some multistep formulas stiffly stable up to order 11 by a computer search. The stability regions were promising but the methods were not tested so it is not known how they would perform. Formulas are given in Chapter 3 that are apparently A-stable with order 11 and stiffly stable up to order 23 (BLOCK3 formulas).

Bickart and Picel (1973) found block linear one step methods stiffly stable with order up to 25. However, the order is equal to the number of future points solved for at each step, so to attain order 25, a block of 25 future points must be solved for simultaneously. Since systems of ODE's are usually being solved, the total dimensionality would be multiplied by 25. This is usually impractical. The problem is caused by ignoring past values. Bickart and Picel (1973) give  $\alpha$ 's for methods up to order 10. These are inferior to the  $\alpha$ 's of the BLOCK2 methods in Chapter 3 (which solve for only two future points).

Enright (1974) discovered a class of linear multistep formulas using second derivatives. The formulas require the evaluation of the derivative of  $f$ , and solve for one future point at each step. Enright's formulas are stiffly stable up to order 9. The stability properties are the same as a family of quadrature methods given in Chapter 4 which do not require evaluation of the derivative of  $f$ .

This study presents several new families of formulas which avoid some of the problems mentioned above. The formulas use past values to obtain high order without simultaneous solution for an excessively large block of future values. Four families of formulas are stiffly stable with order up to 9 and have  $s = 1$ . Families of block multistep formulas with order up to 24 requiring at most four future points are also presented.

### 1.1 Notation and Definitions

The formulas considered here (called collocation formulas) use a polynomial to represent the past information being saved plus the future points being solved for. That is, given  $y_{n-r}, \dots, y_{n-1}$  and assuming we wish to find  $s$  future points  $y_n, \dots, y_{n+s-1}$ , we define a polynomial  $p(t)$  which takes on these values and possibly also collocates derivatives at

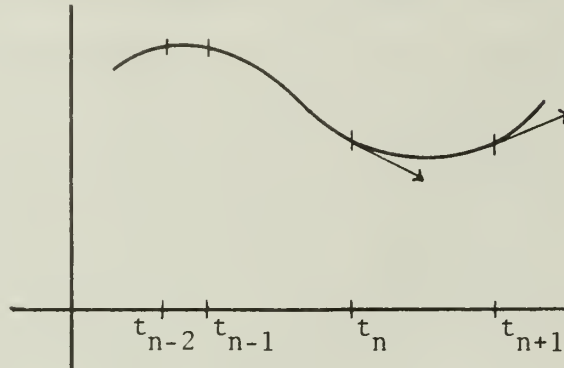


Fig. 1.3. Interpolating polynomial for  $(\theta_0, \theta_1; 1, 1; 2, 2)$

specified points (see Fig. 1.3). We denote a formula as follows:

$$(\theta_0, \theta_1, \dots, \theta_{s-1}; c_{-r}, \dots, c_{-1}; c_0, \dots, c_{s-1}) . \quad (1.3)$$

(The  $\theta_i$  will be explained below.) This notation indicates that the formula finds a polynomial  $p$  such that

$$\left. \begin{aligned} p(t_{n+i}) &= y_{n+i} \\ p'(t_{n+i}) &= y'_{n+i} \\ &\dots \\ p^{(c_i-1)}(t_{n+i}) &= y_{n+i}^{(c_i-1)} \end{aligned} \right\} \quad i = -r, \dots, s-1 ,$$

where

$$y'_{n+i} \equiv f(y_{n+i}, t_{n+i}) , \quad (1.4)$$

$$y_{n+i}^{(j)} \equiv D^{j-1} f(y_{n+i}, t_{n+i}) .$$

Note that the polynomial is a function of  $h_n$  and  $y_{n+i}, t_{n+i}$  for  $i = -r, \dots, s-1$ , which is not explicitly shown in the notation. Let  $C$  denote

$\sum_{i=-r}^{s-1} c_i$ . The polynomial satisfies  $c_i$  conditions at the  $i^{\text{th}}$  point and is

thus of degree at most  $C - 1$ . Because the polynomial is defined using  $c_0 + \dots + c_{s-1}$  pieces of future information, we need  $c_0 + \dots + c_{s-1}$  equations, in addition to the saved values, to define  $p$ . The equations (1.4) above define the higher derivatives at future points in terms of the corresponding future values, so that  $s$  equations are necessary to define  $p$ , one for each future point. Most of the formulas considered here use a collocating condition and require the polynomial to satisfy the differential equation at  $s$  other points:

$$p'(t_n + \theta_i h_n) = f(p(t_n + \theta_i h_n), t_n + \theta_i h_n), \quad i = 0, \dots, s-1. \quad (1.5)$$

For practical reasons we usually require each  $c_i$  to be 1 or 2. Then the formula needs the right hand side of the ODE,  $f$ , but not its derivatives. These formulas have the same desirable stability properties as other formulas which do require the evaluation of  $Df$ . This is important because the exact form of  $Df$  is often unknown or too complicated to calculate and an accurate approximation to  $Df$  is in general too expensive to compute. Because the theory developed below is for arbitrary positive integers  $c_i$ , collocation methods with  $c_i > 2$  can be used to solve ODE's which allow easy computation of  $Df$  (for example, a linear time independent ODE).

An alternative source for the  $s$  additional equations is to require  $p$  to satisfy

$$p(t_n + \theta_i h_n) = p(t_{n-1}) + \int_{t_{n-1}}^{t_n + \theta_i h_n} f(p(\xi), \xi) d\xi, \quad i = 0, \dots, s-1. \quad (1.6)$$

In practice, a quadrature formula is used to replace the integral. The quadrature formula does not affect the stability properties as long as it is exact for polynomials with degree equal to  $C - 1$ . If the integral equations are used then the formula will be denoted

$$I(\theta_0, \dots, \theta_{s-1}; c_{-r}, \dots, c_{-1}; c_0, \dots, c_{s-1}) .$$

The integral formulas are more complicated to implement than the differentiation methods and will only be briefly considered in Chapter 4.

A summary of the properties of some families of formulas discovered is presented in Table 1.1. Here we use the notation  $1^k$  to mean a block of  $k$  successive  $c_i$  all equal to 1.

Table 1.1

## STABILITY SUMMARY

Formula	Maximum Order with alpha > 89°	Maximum Order for Stiff Stability
BACKDIF (0:1 <sup>r</sup> ;1)	2	6
COLLOC1 (1:1 <sup>r</sup> ;2)	4	9
COLLOC2 (1:2,1 <sup>r-1</sup> ;2)	4	9
BLOCK2 (1/2,3/2:1 <sup>r</sup> ;2,2)	9	17
BLOCK3 (1/2,3/2,5/2:1 <sup>r</sup> ;2,2,2)	15	23
BLOCK4 (1/2,3/2,5/2,7/2:1 <sup>r</sup> ;2,2,2,2)	20	24
INTEG1 I(0:1 <sup>r</sup> ;2)	4	9
INTEG2 I(0:2,1 <sup>r-1</sup> ;2)	4	9

As an example of the above notation, Gear's formulas (1969) can be written as the class of formulas

$$(0:1^r;1) , r = 1, \dots, 6 .$$

This means that the formula finds a polynomial  $p$  which interpolates  $r$  past points and 1 future point:

$$p(t_{n+i}) = y_{n+i} , \quad i = -r, \dots, 0$$

and the future point is chosen to satisfy

$$p'(t_n) = f(p(t_n), t_n) ,$$

that is,

$$p'(t_n) = f(y_n, t_n) ,$$

so

$$h_n p'(t_n) - h_n y'_n = 0 .$$

These formulas are  $A(\alpha)$ -stable and stiffly stable with  $\alpha$  and  $D$  as in Table 1.2. The regions of absolute stability are the exteriors of the curves shown in Fig. 1.4, which is included for comparison with other collocation formulas.

Table 1.2

BACKDIF  $(0:1^r;1)$

$r$	order	$D$	$\alpha(^{\circ})$
1	1	0	90
2	2	0	90
3	3	-0.083	86.0
4	4	-0.667	73.4
5	5	-2.327	51.8
6	6	-6.070	17.9

In order to write the collocating polynomial  $p$  it is convenient to define the basis functions,  $\phi_{jk}$ , associated with the formula (1.3). If  $-r \leq j \leq s-1$ , and  $0 \leq k \leq c_j-1$ , then  $\phi_{jk}$  is defined to be the unique

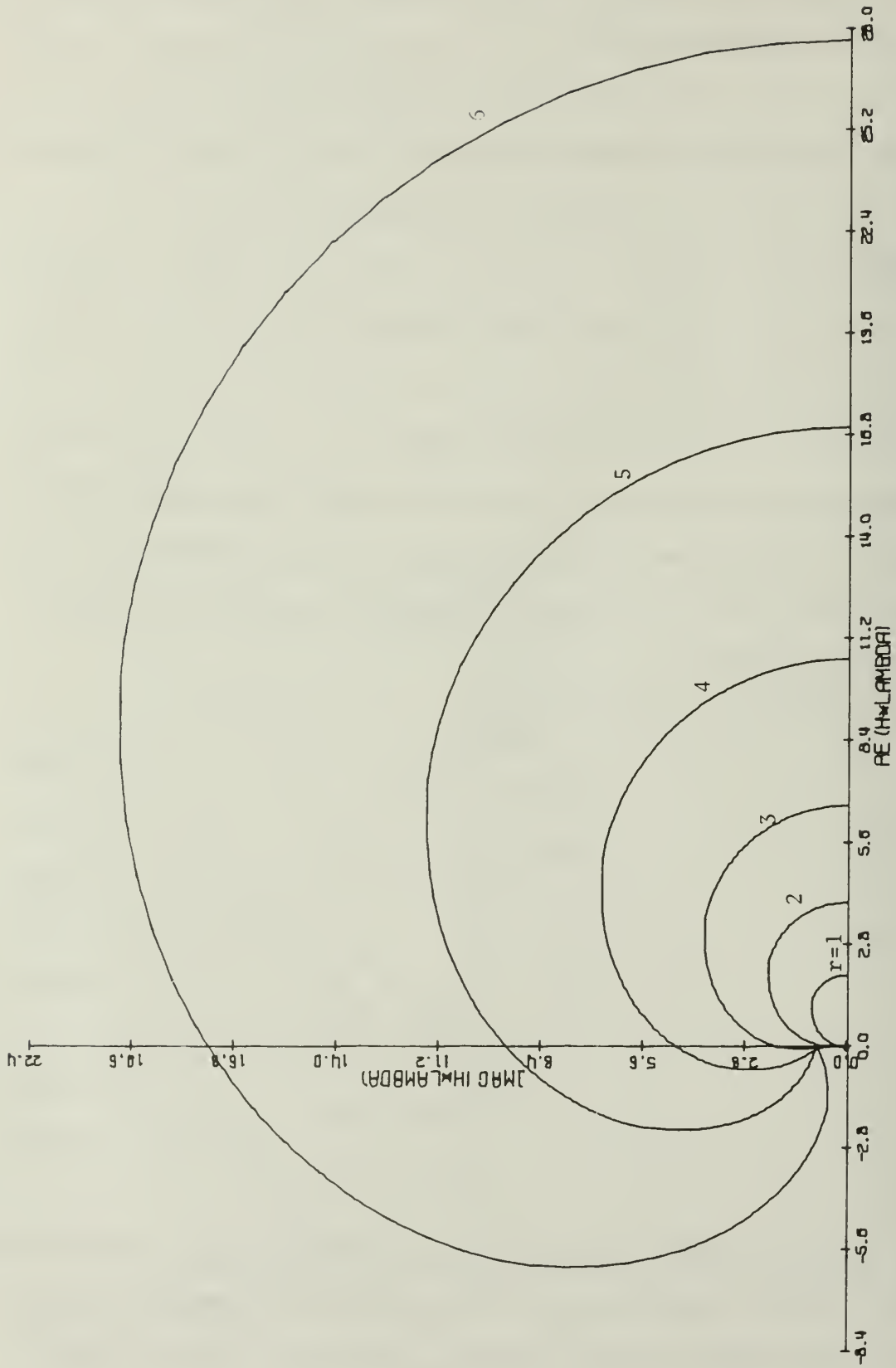


Fig. 1.4. Stability Regions for BACKDIF  $(0:1^r;1)$ ,  $r = 1, \dots, 6$  Order 1-6

polynomial of degree at most  $C - 1$  satisfying

$$\phi_{jk}^{(\ell)} \left( \frac{t_{n+m} - t_n}{h_n} \right) = k! \delta_{jm} \delta_{k\ell} \quad (1.7)$$

for all  $\ell, m$  such that  $-r \leq m \leq s-1$  and  $0 \leq \ell \leq c_m - 1$ . For convenience we define  $\phi_{jk} \equiv 0$  if  $k \geq c_i$ .

Then the interpolation polynomial  $p$  can be written

$$p(t) = \sum_{j=-r}^{s-1} \sum_{k=0}^{c_i-1} \phi_{jk} \left( \frac{t-t_n}{h_n} \right) \frac{h_n^k y_{n+j}^{(k)}}{k!}. \quad (1.8)$$

In the fixed stepsize case with  $h_n = h$ , equation (1.7) assumes the simpler form

$$\phi_{jk}^{(\ell)}(m) = k! \delta_{jm} \delta_{k\ell}, \quad \begin{array}{l} -r \leq m \leq s-1, \\ 0 \leq \ell \leq c_m - 1. \end{array} \quad (1.9)$$

In the rest of the text,  $\sum_{j,k}$  denotes  $\sum_{j=-r}^{s-1} \sum_{k=0}^{c_j-1}$ .

As with linear multistep methods (Gear and Tu, 1974), either an interpolation or variable step technique may be used to change stepsizes. To conform to the nomenclature introduced in that paper, the combination of a set of formulas, the technique used to handle variable stepsizes, and the schemes used to select stepsizes and formulas is called a method. In the variable step technique, the formulas are always used directly as described above with  $\phi_{jk}$  defined as in (1.7). Thus the different stepsizes are taken into account in calculating the interpolating polynomial. In the interpolation stepsize changing technique, the saved information is kept with equal stepsizes  $t_{n+j} - t_{n+j-1} = h_n$ . If the stepsize is changed, an interpolating polynomial is determined by the saved values (including any saved derivatives). The new saved values are

determined by evaluating the interpolating polynomial at the new points.

That is, if the method is saving the values and scaled derivatives

$$y_{n-r}, h_n y'_{n-r}, \dots, \frac{h_n^{(c_{-r}-1)} y_n^{(c_{-r}-1)}}{(c_{-r}-1)!} \text{ at } t_n - r h_n,$$

...

$$y_{n+s-1}, h_n y'_{n+s-1}, \dots, \frac{h_n^{(c_{s-1}-1)} y_{n+s-1}^{(c_{s-1}-1)}}{(c_{s-1}-1)!} \text{ at } t_n + (s-1)h_n,$$

represented by the interpolating polynomial  $p$ , and it is desired to change the stepsize to  $\tilde{h}_n$ , these values are replaced by

$$p(\tilde{t}_{n-r}), \tilde{h}_n p'(\tilde{t}_{n-r}), \dots, \frac{\tilde{h}_n^{(c_{-r}-1)} p^{(c_{-r}-1)}(\tilde{t}_{n-r})}{(c_{-r}-1)!},$$

...

$$p(\tilde{t}_{n+s-1}), \tilde{h}_n p'(\tilde{t}_{n+s-1}), \dots, \frac{\tilde{h}_n^{(c_{s-1}-1)} p^{(c_{s-1}-1)}(\tilde{t}_{n+s-1})}{(c_{s-1}-1)!},$$

where  $\tilde{t}_{n+j} = t_{n+s-1} + (j - s + 1)\tilde{h}_n$ . If the saved information is stored in Nordsieck form (Gear, 1971, p. 148), then the interpolation stepsize changing technique reduces to multiplication by a diagonal matrix. When the saved data is stored at equally spaced times, the basis functions are defined as in (1.9) and are independent of  $h_n$ . Thus the methods reduce to a fixed stepsize method along with the interpolation technique.

## 2. SIMPLE COLLOCATION METHODS

In this chapter we will consider methods using formulas

$$(\theta; c_{-r}, \dots, c_{-1}; c_0) \quad (2.1)$$

satisfying

$$p'(t_n + \theta h_n) = f(p(t_n + \theta h_n), t_n + \theta h_n), \quad (2.2)$$

i.e., methods solving (1.5) with  $s = 1$ . For convenience define

$\alpha_{jk} \equiv \phi'_{jk}(\theta)$ ,  $\beta_{jk} \equiv \phi_{jk}(\theta)$ . Then (2.2) becomes

$$\sum_{jk} \alpha_{jk} \frac{h_n^k y_{n+j}^{(k)}}{k!} = h_n f \left( \sum_{jk} \beta_{jk} \frac{h_n^k y_{n+j}^{(k)}}{k!}, t_n + \theta h_n \right). \quad (2.3)$$

### 2.1 Order

To define the order of the formula (2.1), let  $y_{n+i} = y(t_{n+i})$ ,  $i = -r, \dots, 0$ , where  $y(t)$  is the exact solution of (1.1). Define  $p$  as in (1.8), and let

$$D_n(y) = h_n p'(t_n + \theta h_n) - h_n f(p(t_n + \theta h_n), t_n + \theta h_n)$$

be the amount by which the exact solution fails to satisfy the formula.

Then the formula is defined to be of order  $k$  if

$$D_n(y) = O(h^{k+1})$$

for all solutions  $y \in C^{k+1}[t_0, T]$ . By the properties of interpolating polynomials and the Lipschitz property of  $f$ ,

$$p'(t_n + \theta h_n) = y'(t_n + \theta h_n) + O(h^{C-1}),$$

$$\begin{aligned} f(p(t_n + \theta h_n), t_n + \theta h_n) &= f(y(t_n + \theta h_n) + O(h^C), t_n + \theta h_n) \\ &= y'(t_n + \theta h_n) + O(h^C). \end{aligned}$$

Thus in general the order of the formula is  $C - 1$ , and if  $\theta$  is chosen

such that

$$p'(t_n + \theta h_n) = y'(t_n + \theta h_n) + O(h_n^C),$$

the formula is of order  $C$ .

Example 2.1  $(-1/2; 2; 2)$

The Hermite basis functions for this formula are

$$\phi_{-1,0}(\tau) = 2\tau^3 + 3\tau^2$$

$$\phi_{-1,1}(\tau) = \tau^3 + \tau^2$$

$$\phi_{0,0}(\tau) = -2\tau^3 - 3\tau^2 + 1$$

$$\phi_{0,1}(\tau) = \tau^3 + 2\tau^2 + \tau.$$

With  $\theta = -1/2$ , we find

$$\alpha_{-1,0} = -3/2 \quad \beta_{-1,0} = 1/2$$

$$\alpha_{-1,1} = -1/4 \quad \beta_{-1,1} = 1/8$$

$$\alpha_{0,0} = 3/2 \quad \beta_{0,0} = 1/2$$

$$\alpha_{0,1} = -1/4 \quad \beta_{0,1} = -1/8.$$

Thus given  $y_{n-1}$  the formula chooses  $y_n$  to satisfy

$$\begin{aligned} & -3/2 y_{n-1} - 1/4 h_n y'_{n-1} + 3/2 y_n - 1/4 h_n y'_n = \\ & h_n f(1/2 y_{n-1} + 1/8 h_n y'_{n-1} + 1/2 y_n - 1/8 h_n y'_n, t_n - 1/2 h_n). \end{aligned}$$

Because

$$-3/2 y_{n-1} - 1/4 h_n y'_{n-1} + 3/2 y_n - 1/4 h_n y'_n = h_n y'_{n-1/2} + O(h^5)$$

the formula is of order  $4 = \deg(p) + 1$ . If the formula is applied to

$y' = \lambda y$  with  $h_n = h$  we find

$$y_n = \frac{1 + 1/2 \mu + 1/12 \mu^2}{1 - 1/2 \mu + 1/12 \mu^2} y_{n-1}. \quad (2.4)$$

The rational coefficient of  $y_{n-1}$  in (2.4) is the  $[2,2]$  Padé approximant to  $e^\mu$ . Thus the method is A-stable (Birkhoff and Varga, 1965).

Suppose we apply the formula (2.1) to exact past values  $y_{n+i} = y(t_{n+i})$ ,  $i = -r, \dots, -1$ , choosing  $y_n$  so (2.2) is satisfied. Then the local truncation error is defined to be

$$d_n = y_n - y(t_n) . \quad (2.5)$$

A formula was intuitively defined in Chapter 1 to be of order  $k$  if  $d_n = O(h^{k+1})$ . The definition in this chapter, which is technically more convenient to verify, is equivalent to the definition in Chapter 1, provided the coefficients of the method are well behaved.

**Theorem 2.1** A formula (2.1) with  $\alpha_{0,0}$  bounded away from 0, and  $\alpha_{0,k}, \beta_{0,k}$  bounded as  $h \rightarrow 0$  is of order  $k$  if and only if  $d_n = O(h^{k+1})$ .

**Proof:** Let  $p(t)$  be the polynomial of the formula, defined using exact values, and  $\tilde{p}$  be the polynomial using exact past values but with  $y_n$  chosen to satisfy (2.2). Then

$$h_n p'(t_n + \theta h_n) - h_n f(p(t_n + \theta h_n), t_n + \theta h_n) = D_n ,$$

$$h_n \tilde{p}'(t_n + \theta h_n) - h_n f(\tilde{p}(t_n + \theta h_n), t_n + \theta h_n) = 0 ,$$

where  $D_n = O(h^{k+1})$  by the definition of order. Let  $y_{n+i} \equiv y(t_{n+i})$  for  $i < 0$ . Then we can rewrite these last two equations using (2.3) as

$$\sum_{jk} \alpha_{jk} \frac{h_n^k y^{(k)}(t_{n+j})}{k!} - h_n f \left( \sum_{jk} \beta_{jk} \frac{h_n^k y^{(k)}(t_{n+j})}{k!} , t_n + \theta h_n \right) = D_n , \quad (2.6)$$

$$\sum_{jk} \alpha_{jk} \frac{h_n^k y_{n+j}^{(k)}}{k!} - h_n f \left( \sum_{jk} \beta_{jk} \frac{h_n^k y_{n+j}^{(k)}}{k!} , t_n + \theta h_n \right) = 0 . \quad (2.7)$$

Subtracting (2.6) from (2.7) and applying the mean value theorem to find a matrix  $J$  with norm bounded independently of  $h$ , we have

$$\sum_{k=0}^{c_0-1} \alpha_{0,k} \frac{h_n^k}{k!} (y_n^{(k)} - y^{(k)}(t_n)) - h_n J \sum_{k=0}^{c_0-1} \beta_{0,k} \frac{h_n^k}{k!} (y_n^{(k)} - y^{(k)}(t_n)) + D_n = 0 . \quad (2.8)$$

If we assume that the first  $c_0-1$  derivatives of  $f$  are bounded, then

$$\begin{aligned} y_n^{(k)} - y^{(k)}(t_n) &= D^{k-1}f(y_n, t_n) - D^{k-1}f(y(t_n), t_n) \\ &= O(\|y_n - y(t_n)\|) , \quad k = 1, \dots, c_0-1 . \end{aligned}$$

Thus

$$\alpha_{0,0} d_n = -D_n + O(h \|d_n\|) .$$

Because  $\alpha_{0,0}$  is bounded away from 0 as  $h \rightarrow 0$ ,

$$d_n = \frac{-D_n}{\alpha_{0,0}} + O(h \|d_n\|) ,$$

and if we assume the formula is order  $k$ , then

$$d_n = O(h^{k+1}) .$$

Conversely, suppose  $d_n = O(h^{k+1})$ . Because  $\alpha_{0,k}$  and  $\beta_{0,k}$  are bounded as  $h \rightarrow 0$ , it follows from (2.8) that

$$D_n = O(h^{k+1}) .$$

Thus the formula is of order  $k$ .

Q.E.D.

To show conditions under which the hypotheses of the theorem are satisfied, we need the next lemma. First we introduce stepsize change variables

$$\delta_i = \frac{h_i - h_{i-1}}{h_{i-1}} .$$

Lemma 2.1 The coefficients  $\alpha_{jk}$  and  $\beta_{jk}$  can be represented in the form

$$\alpha_{jk} = \alpha_{jk}^0 + R_{jk}(\delta_{n-r+2}, \dots, \delta_n) ,$$

$$\beta_{jk} = \beta_{jk}^0 + Q_{jk}(\delta_{n-r+2}, \dots, \delta_n) ,$$

where  $\alpha_{jk}^0, \beta_{jk}^0$  are the coefficients of the fixed stepsize formula and  $R_{jk}, Q_{jk}$  are rational functions in  $\delta_{n-r+2}, \dots, \delta_n$  such that  $R_{jk}(0, \dots, 0) = Q_{jk}(0, \dots, 0) = 0$ .

Proof: It is enough to show that the Hermite basis function  $\phi_{jk}$  can be written as a polynomial with coefficients that are rational in  $\delta_i$ , i.e.,

$$\phi_{jk} = \sum_{i=0}^{C-1} q_i(\delta_{n-r+2}, \dots, \delta_n) t^i \quad (2.9)$$

where  $q_i$  is a rational function. Then

$$\alpha_{jk} = \phi'_{jk}(\theta) = \sum_{i=0}^{C-1} i q_i(\delta_{n-r+2}, \dots, \delta_n) \theta^{i-1}$$

and

$$\beta_{jk} = \phi_{jk}(\theta) = \sum_{i=0}^{C-1} q_i(\delta_{n-r+2}, \dots, \delta_n) \theta^i.$$

Thus  $\alpha_{jk}$  and  $\beta_{jk}$  are rational functions of  $\delta_i$  and we arrive at the lemma by defining

$$R_{jk}(\delta_{n-r+2}, \dots, \delta_n) = \sum_{i=0}^{C-1} i [q_i(\delta_{n-r+2}, \dots, \delta_n) - q_i(0, \dots, 0)] \theta^{i-1}$$

and

$$Q_{jk}(\delta_{n-r+2}, \dots, \delta_n) = \sum_{i=0}^{C-1} [q_i(\delta_{n-r+2}, \dots, \delta_n) - q_i(0, \dots, 0)] \theta^i.$$

The representation (2.9) is trivial if  $\phi_{jk} \equiv 0$ , so assume that  $-r \leq j \leq 0$  and  $0 \leq k \leq c_j - 1$ . From (1.7)  $\phi_{jk}$  is defined to be the unique polynomial of degree at most  $C - 1$  satisfying

$$\phi_{jk}^{(\ell)} \left( \frac{t_{n+m} - t_n}{h_n} \right) = k! \delta_{jm} \delta_{k\ell} \quad (2.10)$$

for all  $\ell, m$  such that  $-r \leq m \leq 0$  and  $0 \leq \ell \leq c_{m-1}$ . Let

$$\phi_{jk} = \sum_{i=0}^{C-1} q_i t^i.$$

Define

$$\omega_m = \frac{t_{n+m} - t_n}{h_n} = - \sum_{i=m+1}^0 \frac{h_{n+i}}{h_n}.$$

Then (2.10) becomes

$$\begin{pmatrix} 1 & \omega_{-r} & \omega_{-r}^2 & \dots & \omega_{-r}^{C-1} \\ 0 & 1 & 2\omega_{-r} & \dots & (C-1)\omega_{-r}^{C-2} \\ & & & \dots & \\ 1 & \omega_0 & \omega_0^2 & \dots & \omega_0^{C-1} \\ & & & \dots & \\ 0 & 0 & 0 & \dots & \frac{(C-1)!}{(C-c_0)!} \omega_0^{C-c_0} \end{pmatrix} \begin{pmatrix} q_0 \\ q_1 \\ \dots \\ q_{C-1} \end{pmatrix} = \begin{pmatrix} \delta_{j,-r} \delta_{k,0} \\ \delta_{j,-r} \delta_{k,1} \\ \dots \\ \delta_{j,0} \delta_{k,0} \\ \dots \\ (c_0-1)! \delta_{j,0} \delta_{k,c_0-1} \end{pmatrix} \quad (2.11)$$

Because  $\omega_i$  are distinct, the determinant of the matrix is nonzero. This follows from the uniqueness of Hermite interpolatory polynomials (Davis, 1963).

Now

$$\frac{h_{n+i}}{h_n} = \frac{-1}{\prod_{j=i}^{n-1} h_{n+j}} = \frac{-1}{\prod_{j=i}^{n-1} \frac{1}{1+\delta_{n+j+1}}}.$$

It follows that  $\omega_{-r}, \dots, \omega_0$  are all rational functions of  $\delta_{n-r+2}, \dots, \delta_n$  and since the rational functions in  $r-1$  variables form a field, the  $q_i$  are also rational function in  $\delta_{n-r+2}, \dots, \delta_n$ . Q.E.D.

We introduce the usual stepsize selection scheme and assume the stepsize is given by

$$h_n = h\eta(t_n, h) \quad (2.12)$$

where for all  $h > 0$ ,  $t_0 \leq t \leq T$ ,

$$1 \geq \eta(t, h) \geq \Delta > 0.$$

Then

$$\delta_i = \frac{h_i - h_{i-1}}{h_{i-1}} = \frac{\eta(t_i, h) - \eta(t_{i-1}, h)}{\eta(t_{i-1}, h)}$$

satisfies  $1/\Delta - 1 \geq \delta_i \geq \Delta - 1 > -1$ .

Because the matrix in (2.11) is nonsingular whenever  $\delta_{n-r+2}, \dots, \delta_n$  are all greater than  $-1$ , the rational functions appearing in Lemma 2.1 are continuous on the domain

$$\{(\delta_{n-r+2}, \dots, \delta_n) \mid \delta_{n-r+2} > -1, \dots, \delta_n > -1\}.$$

The stepsize selection scheme restricts  $\delta_i$  to lie in a compact subset of this domain.

Corollary 2.1 If a method is described by (2.1) with stepsize selection scheme (2.12), then the coefficients  $\alpha_{jk}, \beta_{jk}$  are bounded for all  $h, n$ .

The remaining hypothesis in Theorem 2.1 requires a further restriction on the stepsize selection scheme, as is illustrated in the following example.

Example 2.2  $(\theta:1,1;1)$  with variable step technique.

Recall  $\delta_n = (h_n - h_{n-1})/h_{n-1}$ . With this notation the basis functions can be written

$$\begin{aligned}\phi_{-2,0}(\tau) &= \frac{(\delta_n + 1)^2}{\delta_n + 2} \tau(\tau + 1), \\ \phi_{-1,0}(\tau) &= -(\delta_n + 1)\tau^2 - (\delta_n + 2)\tau, \\ \phi_{0,0}(\tau) &= \frac{\delta_n + 1}{\delta_n + 2} \tau^2 + \frac{2\delta_n + 3}{\delta_n + 2} \tau + 1.\end{aligned}$$

Thus

$$\begin{aligned}\alpha_{0,0} &= \phi'_{0,0}(\theta) = 2\theta \frac{\delta_n + 1}{\delta_n + 2} + \frac{2\delta_n + 3}{\delta_n + 2}, \\ \alpha_{0,0}^0 &= \theta + 3/2.\end{aligned}$$

We see that  $\alpha_{0,0}^0 \neq 0$  unless  $\theta = -3/2$ . On the other hand,  $\alpha_{0,0}$  can be 0 if and only if  $\delta_n$  satisfies

$$\delta_n = -\frac{\theta + 3/2}{\theta + 1}. \quad (2.13)$$

If  $\theta > -1$ , then (2.13) shows  $\delta_n < -1$  to make  $\alpha_{0,0} = 0$ . Since  $\delta_n > -1$  for all stepsizes, we see that  $\alpha_{0,0} \neq 0$  for all stepsize selection schemes as long as  $\theta > -1$ .

If  $\theta < -1$  and  $\alpha_{0,0}^0 \neq 0$ , then by continuity  $\alpha_{0,0} \neq 0$  for  $\delta_n$  sufficiently small, and we can apply Theorem 2.1 for a stepsize selection scheme that satisfies  $\delta_n = O(h)$  as  $h \rightarrow 0$ .

The behavior of the method in the example is typical of collocation methods.

**Theorem 2.2** Suppose a method described by (2.1) with stepsize selection scheme (2.12) has  $\alpha_{0,0} \neq 0$  when the stepsize is constant and one of the following conditions is satisfied:

- 1)  $\theta > 0$ ,
- 2)  $\delta_i = O(h)$ , for all  $i$ .

Then the method is of order  $k$  if and only if  $d_n = O(h^{k+1})$ .

**Proof:** By Corollary 2.1 and Theorem 2.1, it is enough to show  $\alpha_{0,0}$  is bounded away from 0 as  $h \rightarrow 0$ .

If  $\delta_i = O(h)$ , then the continuity of  $\alpha_{0,0}$  as a function of  $\delta_i$  shows  $\alpha_{0,0}$  is bounded away from 0 for sufficiently small  $h$ .

Suppose  $\theta > 0$ . We need to examine the structure of  $\phi_{0,0}$ , since  $\alpha_{0,0} = \phi'_{0,0}(\theta)$ . Let

$$\tau_0 = 0 \leq \tau_i \leq \dots \leq \tau_{C-1}$$

be the abscissas  $\{(t_{n+m} - t_n)/h_n : m = -r, \dots, 0\}$  with the  $m^{\text{th}}$  abscissa occurring  $c_m$  times. Then using divided differences,  $a_i$ , we can write

$$\phi_{0,0}(\tau) = \sum_{i=1}^{C-1} a_i \prod_{j=0}^{i-1} (\tau - \tau_j)$$

and thus

$$\phi'_{0,0}(\tau) = \sum_{i=1}^{C-1} a_i \left( \prod_{j=0}^{i-1} (\tau - \tau_j) \right)',$$

It is clear from the structure of the divided difference table that the  $\alpha_i$  for  $i > 0$  have the same sign. Rather than prove this formally we illustrate by an example which shows the signs as +, -, or 0:

$\tau_4$	0			
		0		
$\tau_3$	0	0		
		0	+	
$\tau_2$	0	+	-	
		+	-	
0	1	-		
		0		
0	1			

The lower diagonal gives  $a_i$ , so  $a_0 = 1$ ,  $a_1 = 0$ , and  $a_2, a_3, a_4$  are all less than 0.

The derivative

$$\left( \prod_{j=0}^{i-1} (\tau - \tau_j) \right)' = \sum_{m=0}^{i-1} (\tau - \tau_m)^{-1} \prod_{j=0}^{i-1} (\tau - \tau_j)$$

is a sum of products of  $(\tau - \tau_j)$ . Thus if  $\tau = \theta > 0$  these derivatives are all positive. Hence

$$\begin{aligned} |\alpha_{0,0}| &= \sum_{i=1}^{C-1} |a_i| \sum_{m=0}^{i-1} (\theta - \tau_m)^{-1} \prod_{j=0}^{i-1} (\theta - \tau_j) \\ &\geq \sum_{i=1}^{C-1} |a_i| i \theta^{i-1} \\ &> 0 \end{aligned}$$

and  $\alpha_{0,0}$  is bounded away from 0 independently of  $h$ .

Q.E.D.

## 2.2 Stability Regions

To find the absolute stability region of formula (2.1), we apply it to  $y' = \lambda y$ , with constant stepsize  $h$ . Then (2.3) becomes

$$\sum_{jk} \alpha_{jk} \frac{(h\lambda)^k y_{n+j}}{k!} = h\lambda \left( \sum_{jk} \beta_{jk} \frac{(h\lambda)^k y_{n+j}}{k!} \right), \quad (2.14)$$

where the  $\alpha_{jk}$ ,  $\beta_{jk}$  are defined by (1.9) and are independent of  $h$ ,  $\lambda$ .

Writing  $\mu \equiv h\lambda$  and collecting terms, we see that (2.14) takes the form

$$a_{-r} y_{n-r} + \dots + a_0 y_n = 0 \quad (2.15)$$

where

$$a_j = \sum_{k=0}^{c_j-1} \left( \alpha_{jk} \frac{\mu^k}{k!} - \beta_{jk} \frac{\mu^{k+1}}{k!} \right).$$

Solutions of (2.15) are

$$y_n = \sum_{i=1}^r d_i \xi_i^n$$

if the  $\xi_i$  are distinct roots of the stability polynomial

$$\pi(\mu, \xi) = a_{-r} + a_{-r+1} \xi + \dots + a_0 \xi^r.$$

If  $\pi$  has multiple roots, then the solutions will include terms of the form  $n^k \xi_i^n$  with  $k > 0$ , which will tend to 0 if and only if  $|\xi_i| < 1$ . Thus the region of absolute stability is

$$\{\mu \mid \text{all roots, } \xi, \text{ of } \pi(\mu, \xi) \text{ have } |\xi| \leq 1 \\ \text{and roots of modulus 1 are simple}\}.$$

For stiff stability we are interested in the component of the absolute stability region connected to  $\mu = \infty$ . The formulas whose stability regions are shown in this paper are all strongly stable at  $\mu = \infty$ , that is, the roots of  $\pi(\infty, \xi)$  are all less than 1 in magnitude. The set of roots of a polynomial is a continuous function of its coefficients, so such formulas are strongly stable in the exteriors of the curves

$\{\mu \mid \text{there exists } \xi \text{ with } |\xi| = 1 \text{ such that } \pi(\mu, \xi) = 0\}.$

These loci can be plotted by regarding the stability polynomial as a polynomial in  $\mu$  with coefficients which are functions of  $\xi$  and plotting the solutions of  $\mu$  for  $\xi = e^{i\omega}$ ,  $\omega \in [0, 2\pi]$ .

Example 2.3  $(1:l^r;2)$ ,  $r = 1, \dots, 8$

The stability polynomials have the form

$$\begin{aligned} \pi(\mu, \xi) = & (\alpha_{-r,0} - \beta_{-r,0}\mu) + \dots + (\alpha_{-1,0} - \beta_{-1,0}\mu)\xi^{r-1} \\ & + (\alpha_{0,0} + (\alpha_{0,1} - \beta_{0,0})\mu - \beta_{0,1}\mu^2)\xi^r. \end{aligned}$$

As  $\mu \rightarrow \infty$  the roots  $\xi$  approach the roots of the polynomial

$$\xi^r = 0$$

and so are less than 1 in magnitude. Hence the formulas are strongly stable at  $\infty$ .

The results of Section 2.1 show that the order of  $(1:l^r;2)$  is  $r + 1$ , and the stability regions of Figs. 2.1 and 2.2 show empirically that the formulas are stiffly stable for  $r \leq 8$ . The corresponding  $D$ 's and  $\alpha$ 's are shown in Table 2.1. Note that the stability regions are much better than those of the backward differentiation formulas show in Fig. 1.4. This family of formulas has been implemented as a variable stepsize, variable order package using the interpolation stepsize changing technique, and is further discussed in Section 2.3.

Table 2.1  
COLLOC1  $(1:l^r;2)$

r	order	D	$\alpha(^{\circ})$	r	order	D	$\alpha(^{\circ})$
1	2	0	90	5	6	-0.237	83.1
2	3	0	90	6	7	-0.820	72.5
3	4	0	90	7	8	-2.079	55.6
4	5	-0.028	88.7	8	9	-4.554	29.5

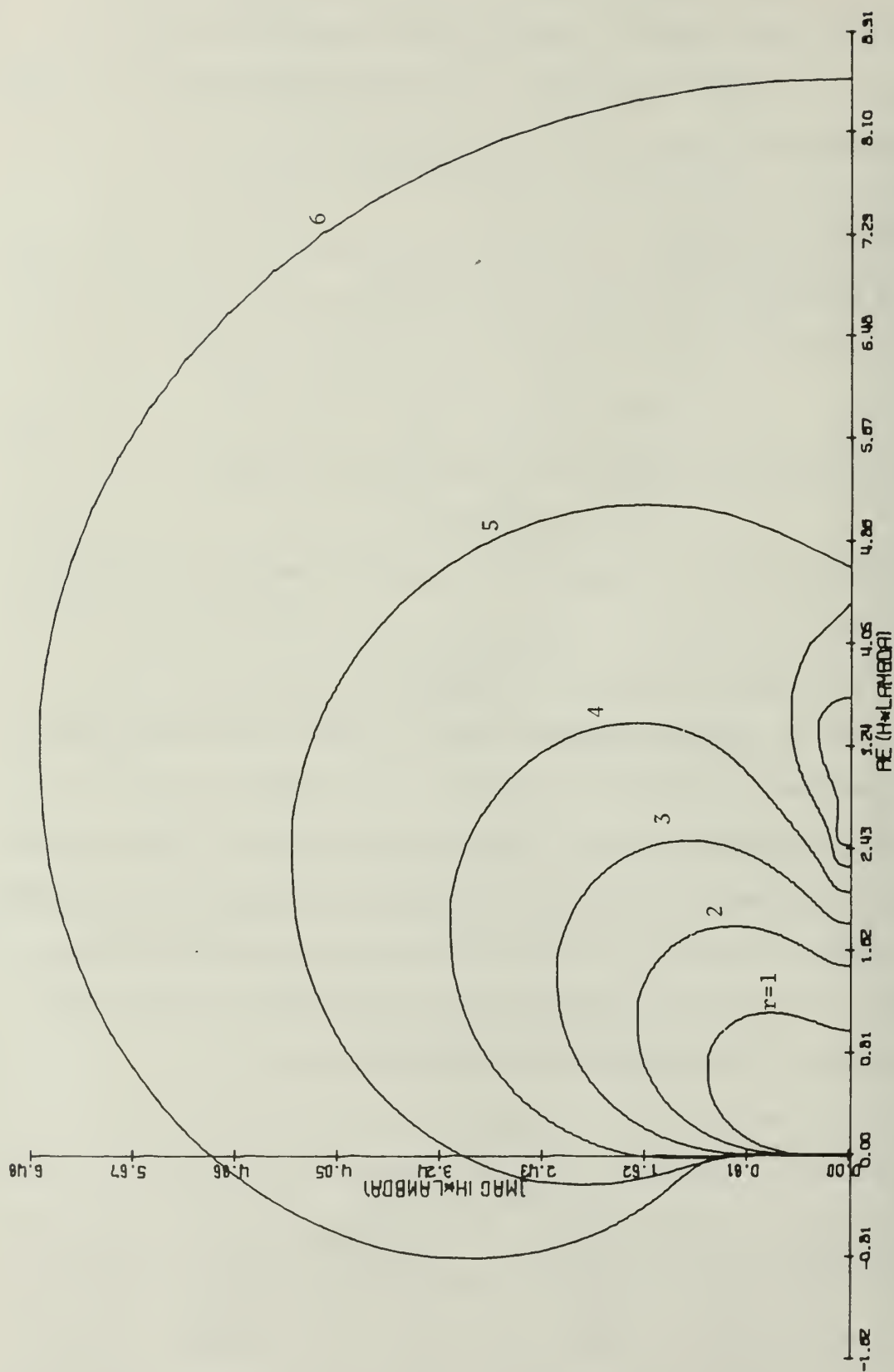


Fig. 2.1. COLLOC1  $(1:1^r; 2)$ ,  $r = 1, \dots, 6$  Order 2-7

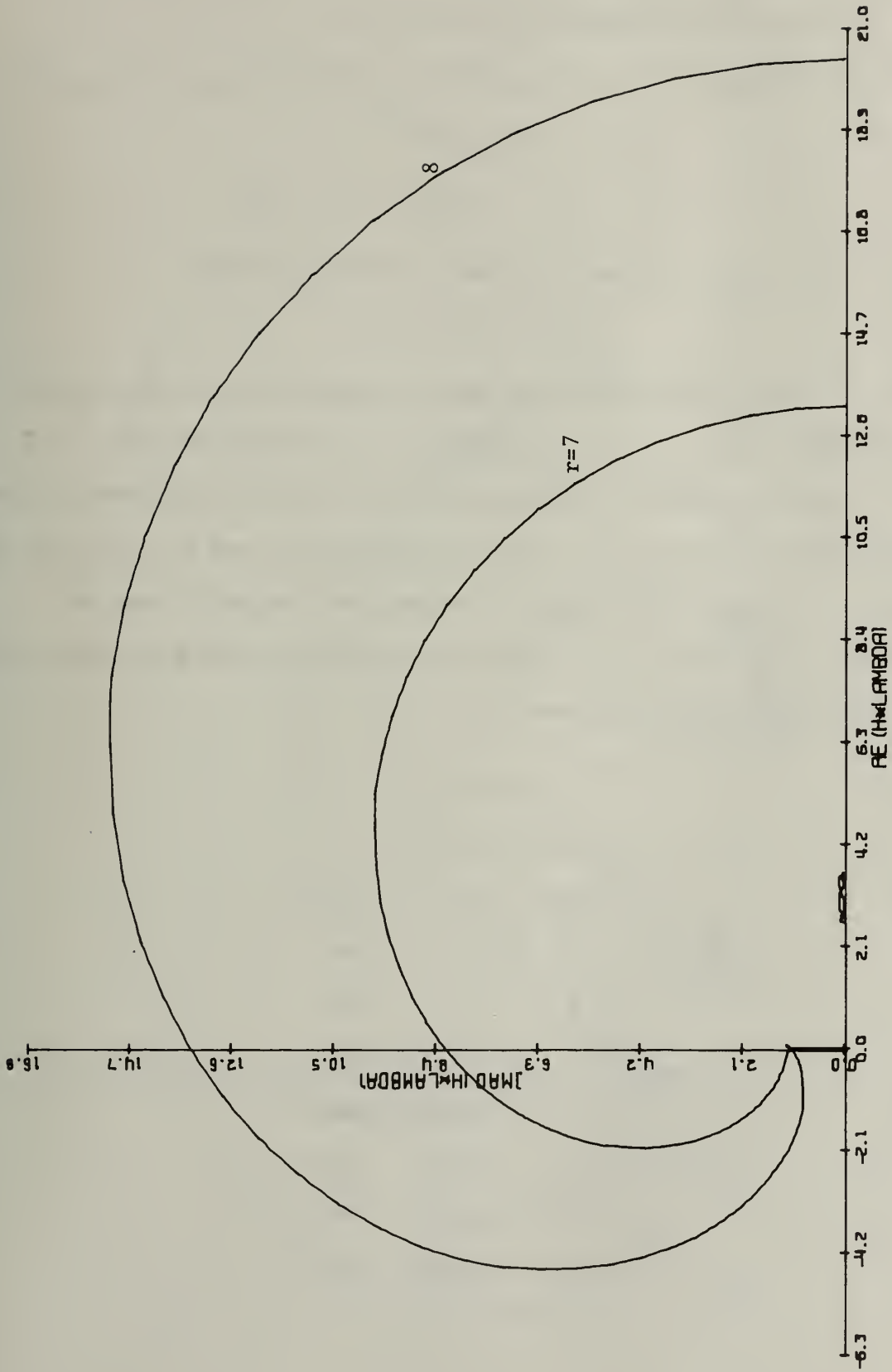


Fig. 2.2. COLLOC1  $(1:1^r; 2)$ ,  $r = 7, 8$  Order 8, 9

Example 2.4  $(1:2, 1^{r-1}; 2)$ ,  $r = 1, \dots, 7$

The stability polynomials have the form

$$\begin{aligned} \pi(\mu, \xi) = & [\alpha_{-r,0} + (\alpha_{-r,1} - \beta_{-r,0})\mu - \beta_{-r,1}\mu^2] + (\alpha_{-r+1,0} - \beta_{-r+1,0}\mu)\xi \\ & + \dots + (\alpha_{-1,0} - \beta_{-1,0}\mu)\xi^{r-1} \\ & + [\alpha_{0,0} + (\alpha_{0,1} - \beta_{0,0})\mu - \beta_{0,1}\mu^2]\xi^r. \end{aligned}$$

As  $\mu \rightarrow \infty$  the roots  $\xi$  approach the roots of the polynomial

$$\beta_{0,1}\xi^r + \beta_{-r-1} = 0.$$

For  $r \leq 7$  these roots are all less than 1 in magnitude, and the formulas are strongly stable at  $\infty$ , as in Example 2.3. The order of  $(1:2, 1^{r-1}; 2)$  is  $r + 2$  and the stability regions in Fig. 2.3 show that the formulas are stiffly stable for order  $\leq 9$ . The corresponding D's and  $\alpha$ 's are shown in Table 2.2. Although the stability regions are similar to those of Example 2.3, the need for the derivative at the last past point makes them much less attractive to implement.

Table 2.2

COLLOC2 $(1:2, 1^{r-1}; 2)$			
<u>r</u>	<u>order</u>	<u>D</u>	<u><math>\alpha(^{\circ})</math></u>
1	3	0	90
2	4	0	90
3	5	-0.038	88.1
4	6	-0.334	80.1
5	7	-1.119	66.2
6	8	-2.742	46.8
7	9	-5.809	21.9

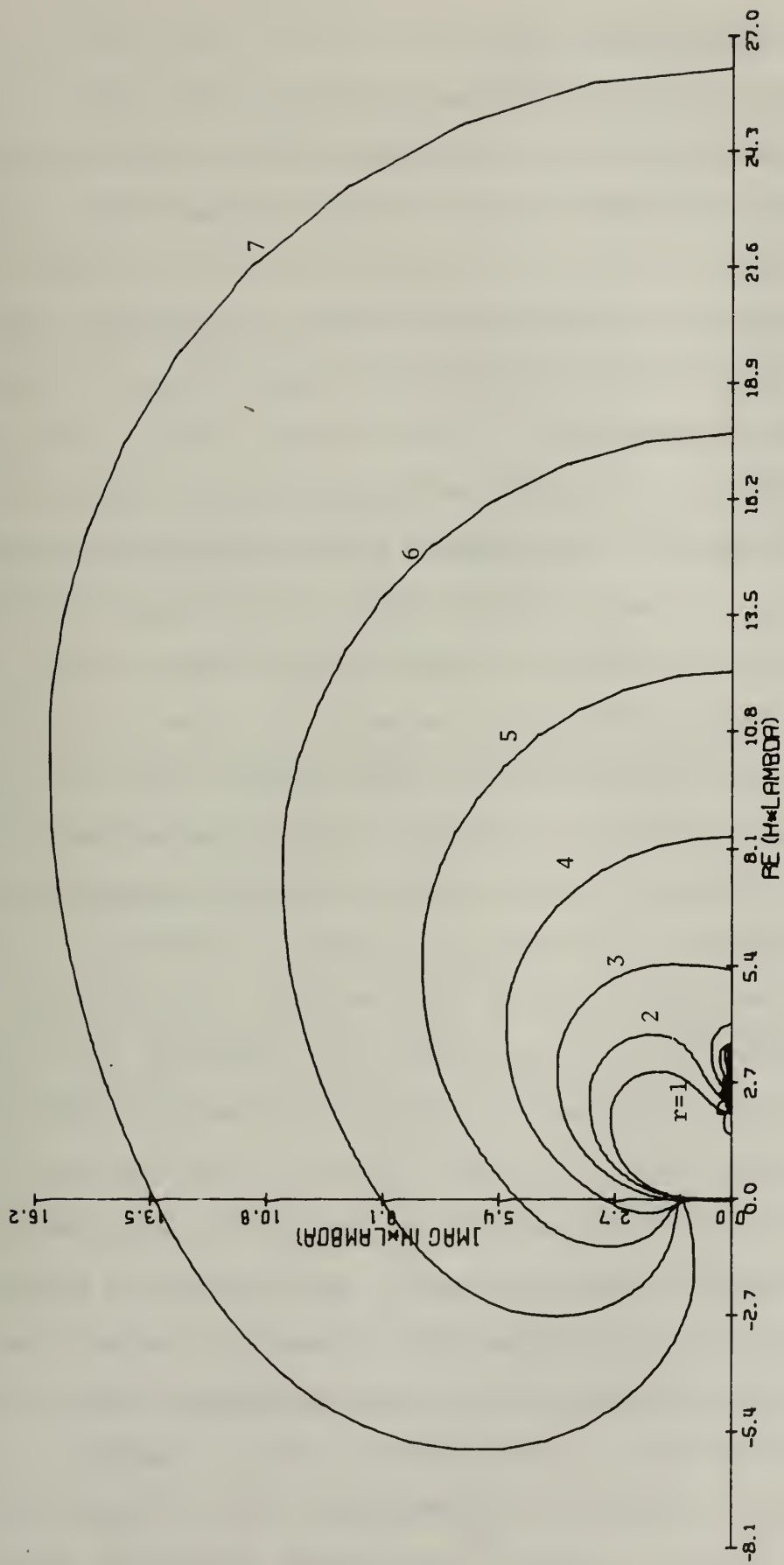


Fig. 2.3. COLLOC2  $(1:2, 1^{r-1}; 2)$ ,  $r = 1, \dots, 7$  Order 3-9

### 2.3 Implementation Considerations

In this section we discuss several topics necessary in writing a computer program which implements the formulas described in this chapter. Computer codes need to handle several practical problems. These include data representation, choice of stepsize and order to achieve a user specified error tolerance, approximate solution of the nonlinear equation (2.2), and the use of the solution of (2.2) to move the solution of the ODE forward by one time step.

For simplicity, we consider the specific family of formulas described in Example 2.3. (See Appendix I for a program which implements these methods.) If other formulas are used, then care may be necessary in choosing the data representation and the stepsize changing technique. For example, the formula  $(\theta:2, l^{r-1}; 2)$  requires  $y'$  at the last past point. If the variable step technique is used to change stepsizes, then either this derivative can be recalculated when it is needed, or all derivatives can be saved. The latter action would reduce the number of function evaluations at the expense of almost doubling the storage requirements. If the interpolation technique is used to change stepsizes, the saved data is usually transformed to the Nordsieck form, which represents all the saved data by the scaled derivatives at a single point of the polynomial interpolating the saved data. If the saved data includes only the values used in the formula  $(\theta:2, l^{r-1}; 2)$ , then when the solution is moved ahead at each step, the component of the data due to  $y'_{n-r}$  must be explicitly calculated and added to each data value of the Nordsieck vector. This additional calculation reduces the convenience of using the Nordsieck form. If all the derivatives are saved, and the Nordsieck vector has length  $2(r + 1)$ , then recalculation of derivatives is unnecessary but the storage requirement is almost doubled. Several of the formulas considered in this

report have the property that the  $c_i$ 's increase monotonically with  $i$ . In this case, only the data values used in the formula need be saved. Similar considerations apply to formula changing. The saved data must be such that the past values needed by the new formula are available and correctly spaced in time.

The implementation of the formulas of Example 2.3 uses the interpolation technique. When used with the Nordsieck form of data representation, stepsize changing and prediction of the next solution value can be efficiently done (Gear, 1971, p. 148). At the  $n^{\text{th}}$  step, the data defining the polynomial (1.8) is saved and is represented by the value and  $r + 1$  derivatives at  $t_n$ . Thus rather than saving

$$\underline{y}_n = (y_n, h_n y'_n, y_{n-1}, \dots, y_{n-r})^T,$$

we save

$$\underline{a}_n = \left( y_n, h_n y'_n, \frac{h_n^2}{2} y''_n, \dots, \frac{h_n^{(r+1)}}{(r+1)!} y_n^{(r+1)} \right)^T.$$

This is possible because the  $c_i$ 's increase monotonically as was mentioned above. In this notation

$$y_n^{(i)} = p^{(i)}(t_n)$$

where  $p$  is the polynomial interpolating  $\underline{y}_n$ . Thus the representation is a linear transformation of the natural representation

$$\underline{a}_n = T \underline{y}_n. \quad (2.16)$$

The equation which defined  $y_n, h_n y'_n$  given  $\underline{a}_{n-1}$  is generally nonlinear and is dependent on the accuracy of an approximate value,  $y_n^p$ , for its ease of solution. To obtain  $y_n^p$  we evaluate the defining polynomial given for  $t_{n-1}$  at  $t_n$ . With our data representation, this is equivalent to multiplying by the Pascal matrix,

$$\underline{a}_n^p = P \underline{a}_{n-1} \quad (2.17)$$

where

$$P = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 0 & 1 & 2 & \dots & r+1 \\ 0 & 0 & 1 & \dots & \begin{pmatrix} r+1 \\ 2 \end{pmatrix} \\ & & & \dots & \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix}$$

is the matrix whose  $(i, j)^{\text{th}}$  entry is  $\begin{pmatrix} j \\ i \end{pmatrix}$ ,  $i, j = 0, \dots, r+1$ .

With this data representation, equation (2.2), which defines  $y_n$ , can be easily written. Let  $\underline{e}_i$  be the unit vector with  $i^{\text{th}}$  element 1. Then (2.2) can be written

$$\underline{e}_1 \cdot \underline{P} \underline{a}_{-n} = f(\underline{e}_0 \cdot \underline{P} \underline{a}_{-n}, t_n + h_n) . \quad (2.18)$$

Thus the polynomial interpolating saved values need not be explicitly found. Values of  $\theta$  other than 1 could be handled by replacing  $\underline{a}_{-n}$  by  $C(\theta)\underline{a}_{-n}$  where  $C(\theta)$  is the matrix with elements  $c_{ij} = \delta_{ij}\theta^i$ , for  $i, j = 0, \dots, r+1$ .

To solve (2.18) we need the first two elements of  $\underline{P} \underline{a}_{-n}$  as functions of the unknowns  $y_n$  and  $h_n y'_n$ . To find the elements of  $\underline{a}_{-n}$  after  $y_n$  and  $h_n y'_n$  are found we need an expression for it also. Let  $P_1$  be the first two rows of  $P$ , and let  $T_1$  be the first two columns of  $T$ . Then using (2.16) and (2.17), it follows that

$$\underline{a}_{-n} = \underline{a}_{-n}^p + T_1 \begin{pmatrix} y_n - \underline{e}_0 \cdot \underline{a}_{-n}^p \\ h_n y'_n - \underline{e}_1 \cdot \underline{a}_{-n}^p \end{pmatrix} , \quad (2.19)$$

$$P_1 \underline{a}_{-n} = P_1 \underline{a}_{-n}^p + P_1 T_1 \begin{pmatrix} y_n - \underline{e}_0 \cdot \underline{a}_{-n}^p \\ h_n y'_n - \underline{e}_1 \cdot \underline{a}_{-n}^p \end{pmatrix} . \quad (2.20)$$

The program in Appendix I uses the  $2 \times 2$  matrix  $P_1 T_1$  to solve (2.18) and the  $(r + 1) \times 2$  matrix  $T_1$  to correct  $\underline{a}_n^P$  for  $\underline{a}_n$ . Note both matrices are independent of stepsize (but do depend on order).

To choose the order and stepsize to use, a code needs to estimate the error committed at the next step as a result of its choices. The program in Appendix I attempts to estimate the local truncation error and keep it less than a user specified tolerance,  $\epsilon$ . Suppose the past values  $y_{n-r}, \dots, y_{n-1}$  are exact and let  $p(t; y_n)$  be the polynomial (1.8) defining the formula with the parameter  $y_n$  explicitly shown. Then with

$$E_n = h_n p'(t_n + \theta h_n; y(t_n)) - h_n f(p(t_n + \theta h_n), t_n + \theta h_n),$$

the local truncation error satisfies

$$d_n = \frac{-E_n}{\alpha_{0,0}} + O(h \|d_n\|),$$

as we saw in Theorem 2.1.

We next estimate  $E_n$ . Let

$$\tau_0 = t_n \geq \tau_1 \geq \dots \geq \tau_{C-1} = t_{n-r}$$

be the abscissas  $\{t_{n+m} : m = -r, \dots, 0\}$  with the  $m^{\text{th}}$  abscissa occurring  $c_m$  times. Let

$$\omega(t) = \prod_{i=0}^{C-1} (t - \tau_i)$$

and let  $y[t]$  be the divided difference (Isaacson and Keller, 1966)

$$y[t] = \int_0^1 \int_0^{t_1} \dots \int_0^{t_{C-1}} y^{(C)} \left( t_C(t - \tau_{C-1}) + \dots + t_1(\tau_1 - \tau_0) + \tau_0 \right) dt_C \dots dt_1.$$

Then the exact solution  $y(t)$  can be written as

$$y(t) = p(t; y(t_n)) + \omega(t)y[t],$$

so that

$$y'(t) = p'(t; y(t_n)) + \omega'(t)y[t] + \omega(t)y'[t] .$$

It is easily seen by induction that if  $|y^{(k)}| \leq M_k$  on an interval containing  $t_{n-r}, \dots, t_n, t$ , then

$$|y^{(k)}[t]| \leq M_{C+k} \frac{k!}{(C+k)!} .$$

In particular, the derivatives of  $y[t]$  are bounded if the appropriate derivatives of  $y(t)$  are.

Since we are using the interpolation technique, the stepsize may be taken as  $h_n$  for the past  $r$  steps. Then

$$\omega(t) = h_n^C \tilde{\omega}\left(\frac{t-t_n}{h_n}\right)$$

and

$$\omega'(t) = h_n^{C-1} \tilde{\omega}'\left(\frac{t-t_n}{h_n}\right)$$

where

$$\tilde{\omega}(\tau) = (\tau + r)^{C-r} \dots (\tau)^{C_0} .$$

Then

$$\begin{aligned} E_n &= h_n y'(t_n + \theta h_n) - h_n^C \tilde{\omega}'(\theta) y[t_n + \theta h_n] - h_n^{C+1} \tilde{\omega}(\theta) y'[t_n + \theta h_n] \\ &\quad - h_n f(y(t_n + \theta h_n)) - h_n^C \tilde{\omega}(\theta) y[t_n + \theta h_n], t_n + \theta h_n) \\ &= h_n y'(t_n + \theta h_n) - h_n^C \tilde{\omega}'(\theta) y[t_n + \theta h_n] + O(h_n^{C+1}) \\ &\quad - h_n y'(t_n + \theta h_n) + h_n f(y(t_n + \theta h_n)) h_n^C \tilde{\omega}(\theta) y[t_n + \theta h_n] \\ &= -h_n^C \tilde{\omega}'(\theta) y[t_n + \theta h_n] + O(h^{C+1}) \\ &= -h_n^C \tilde{\omega}'(\theta) \frac{y^{(C)}(\xi)}{C!} + O(h^{C+1}) , \end{aligned}$$

where  $t_{n-r} \leq \xi \leq t_n + \theta h_n$ . The latter holds because  $y[t_n + \theta h_n] = y^{(C)}(\xi)/C!$ .

However  $y^{(C+1)}$  is bounded, by assumption, so any value of  $\xi$  may be chosen and maintain the equality. The program estimates  $y^C(\xi)$  at some convenient value and uses the above formulas to estimate  $d_n$ . In this case  $C = r + 2$  and the highest approximate derivative saved is  $y_n^{(r+1)}$ , so the desired derivative may be estimated using either the difference  $y_n^{(r+1)} - y_n^{(r+1)}$  or the difference between predicted and corrected values of  $y_n$ , since

$$y_n^p = y(t) + \tilde{\omega}(\theta) \frac{h_n^C y^{(C)}(\xi)}{C!} + O(h^{C+1}) .$$

This is what is done in the program.

The stiff system of equations

$$y' = -By + Uw ,$$

$$y(0) = -U(1,1,1,1)^T ,$$

where  $B = U(\text{diag}(\beta_i))U^*$ ,  $w = (z_1^2)^T$ , and  $z = U^{-1}y$ , was solved using the code in Appendix I and Gear's DIFSUB in two cases. The exact solution is  $Uz$  where

$$z_i = \frac{\beta_i}{1 - (1 + \beta_i) \exp(\beta_i t)} .$$

In case 1,  $\beta_1 = 1000$ ,  $\beta_2 = 800$ ,  $\beta_3 = -10$ ,  $\beta_4 = 0.001$  and

$$U = 1/2 \begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix} .$$

In case 2,  $\beta_1 = 100 + 1000i$ ,  $\beta_2 = \bar{\beta}_1$ ,  $\beta_3 = -10$ ,  $\beta_4 = 0.01$  and

$$U = 1/2 \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ -i & i & 1 & -1 \\ -i & i & -1 & 1 \end{pmatrix}$$

Both methods automatically vary the stepsize to keep the estimated local error less than a given tolerance  $\epsilon$ . Tables 2.3 and 2.4 give the maximum error in the components of  $y$  at the first mesh point after  $t = 10^i$  and the number of function evaluations required to reach that point. In both cases the tolerance  $\epsilon$  is  $10^{-6}$ .

Table 2.3

Comparison for Case 1				
Time	Error	COLLOC	Evaluations	
	DIFSUB		DIFSUB	COLLOC
$1_{10}-3$	$.45_{10}-6$	$.16_{10}-5$	72	101
$1_{10}-2$	$.74_{10}-7$	$.11_{10}-6$	173	229
$1_{10}-1$	$.26_{10}-6$	$.87_{10}-9$	314	377
$1_{10}0$	$.48_{10}-6$	$.15_{10}-6$	448	611
$1_{10}+1$	$.36_{10}-5$	$.92_{10}-6$	591	797
$1_{10}+2$	$.26_{10}-5$	$.60_{10}-7$	693	981
$1_{10}+3$	$.13_{10}-5$	$.38_{10}-6$	777	1161

The collocation program requires more function evaluations than DIFSUB in both cases, if the maximum order in DIFSUB is reduced to 3 in case 2. If DIFSUB is allowed to use its order 4 formula in case 2, it restricts the stepsize and uses more function evaluations than COLLOC.

It is not surprising that DIFSUB takes fewer function evaluations than COLLOC. DIFSUB is a production program which has been extensively tuned and tested, while COLLOC is a test package designed to check whether the formulas developed in this report worked in practice. As a consequence, the code was designed for clarity and ease of debugging rather than efficiency. The efficiency of COLLOC could be increased in several ways. For example, changing the nonlinear equation solver to perform a rank-one update

on the Jacobian,  $J$ , of  $f$  and using  $J$  to calculate the Jacobian of the system (2.18) would save about 20% of the function evaluations used in COLLOC. Observation of Tables 2.3 and 2.4 reveals that COLLOC is solving the equations much more accurately than DIFSUB. Thus tuning of the error estimation scheme would also improve COLLOC relative to DIFSUB. Finally, the error tolerance used in these test problems is rather large, favoring the less accurate DIFSUB. When the tolerance,  $\epsilon$ , is decreased, COLLOC improves relative to DIFSUB.

Table 2.4

Comparison for Case 2

Time	Error			Evaluations		
	DIFSUB <sup>1</sup>	DIFSUB <sup>2</sup>	COLLOC <sup>3</sup>	DIFSUB <sup>1</sup>	DIFSUB <sup>2</sup>	COLLOC <sup>3</sup>
$1_{10}-3$	$.11_{10}-4$	$.69_{10}-5$	$.34_{10}-5$	87	61	105
$1_{10}-2$	$.95_{10}-4$	$.10_{10}-3$	$.33_{10}-4$	349	215	341
$1_{10}-1$	$.25_{10}-4$	$.32_{10}-4$	$.13_{10}-4$	1128	961	1501
$1_{10}0$	$.10_{10}-4$	$.36_{10}-4$	$.13_{10}-5$	1647	2268	2129
$1_{10}+1$	$.20_{10}-4$	$.33_{10}-4$	$.21_{10}-5$	1801	13177	2343
$1_{10}+2$	$.93_{10}-5$		$.14_{10}-5$	1913	--	2529
$1_{10}+3$	$.22_{10}-6$		$.87_{10}-7$	1995	--	2725

<sup>1</sup> maximum order 3<sup>2</sup> maximum order 4<sup>3</sup> maximum order 5

## 3. BLOCK COLLOCATION METHODS

In this chapter we will consider methods using formulas

$$(\theta_0, \dots, \theta_{s-1}; c_{-r}, \dots, c_{-1}; c_0, \dots, c_{s-1}) \quad (3.1)$$

satisfying

$$p'(t_n + \theta_i h_n) = f(p(t_n + \theta_i h_n), t_n + \theta_i h_n), \quad i = 0, \dots, s-1, \quad (3.2)$$

i.e., the most general case of (1.5). For convenience define

$\alpha_{ij}^k \equiv \phi_{jk}'(\theta_j)$ ,  $\beta_{ij}^k \equiv \phi_{jk}(\theta_i)$ . Then (3.2) becomes

$$\sum_{jk} \alpha_{ij}^k \frac{h_n^k y_{n+j}^{(k)}}{k!} = h_n f \left( \sum_{jk} \beta_{ij}^k \frac{h_n^k y_{n+j}^{(k)}}{k!}, t_n + \theta_i h_n \right), \quad i = 0, \dots, s-1. \quad (3.3)$$

Note that in the notation of Chapter 2,  $\alpha_{jk} = \alpha_{0,j}^k$  and  $\beta_{jk} = \beta_{0,j}^k$ . We

will consider the block of  $s$  future points as equally spaced, for notational simplicity. However, the theory in this chapter remains valid when the points are unequally spaced.

3.1 Order

For the theoretical discussion it is easiest to rewrite (3.3) as a vector equation grouping the future points together. Let  $\rho = \lceil r/s \rceil$ ,  $R = \rho s$  and let

$$Y_m = \begin{bmatrix} y_{n+s-1} \\ \dots \\ y_n \end{bmatrix}, \quad Y_{m-1} = \begin{bmatrix} y_{n-1} \\ \dots \\ y_{n-s} \end{bmatrix}, \quad \dots, \quad Y_{m-\rho} = \begin{bmatrix} y_{n-R+s-1} \\ \dots \\ y_{n-R} \end{bmatrix} \quad (3.4)$$

be the solution grouped as  $s$ -vectors, and let  $\tilde{Y}_i$  be the vector with exact values  $y(t_j)$  replacing  $y_j$ .

Define the  $s \times s$  coefficient matrices

$$A_{j,k} = \begin{pmatrix} \alpha_{0,j+s-1}^k & \cdots & \alpha_{0,j}^k \\ \cdots & \cdots & \cdots \\ \alpha_{s-1,j+s-1}^k & \cdots & \alpha_{s-1,j}^k \end{pmatrix}, \quad \begin{matrix} j = -\rho, \dots, 0 \\ k = 0, \dots, \max(c_\ell) - 1 \end{matrix}$$

and

$$B_{j,k} = \begin{pmatrix} \beta_{0,j+s-1}^k & \cdots & \beta_{0,j}^k \\ \cdots & \cdots & \cdots \\ \beta_{s-1,j+s-1}^k & \cdots & \beta_{s-1,j}^k \end{pmatrix}, \quad \begin{matrix} j = -\rho, \dots, 0 \\ k = 0, \dots, \max(c_\ell) - 1 \end{matrix}$$

where  $\alpha_{i,j}^k = \beta_{i,j}^k = 0$  if  $-R \leq j < -r$  or  $k \geq c_j$ . (This amounts to ignoring

the points with  $j$  between  $-R$  and  $-r$  which do not appear in (3.3).) We do not write  $m$  as a function of  $n$  because different formulas may be used to integrate an equation, so the relation between  $m$  and  $n$  may depend on the formula selection scheme. It is understood that  $m$  corresponds to  $n$  in the manner depicted in equations (3.4). Let  $\tilde{c} = \max_j \{c_j\}$ , and let  $\sum'_{jk}$  denote

$\sum_{j=-\rho}^0 \sum_{k=0}^{\tilde{c}}$  for the rest of this chapter. With these definitions, the equation

defining the future point  $Y_m$  can be written as

$$\sum'_{jk} A_{jk} \frac{h_m^{k_Y(k)}}{k!} = h_m f \left( \sum'_{jk} B_{jk} \frac{h_m^{k_Y(k)}}{k!}, t_m + \Theta h_m \right) \quad (3.5)$$

where  $\Theta = (\theta_0, \dots, \theta_{s-1})^T$  and  $f(Y, T)$  means the vector

$$(f(\underline{e}_0 \cdot Y, \underline{e}_0 \cdot T), \dots, f(\underline{e}_{s-1} \cdot Y, \underline{e}_{s-1} \cdot T))^T.$$

With this notation the order of the formula (3.1) may be defined. Let

$y_{n+i} = y(t_{n+i})$ ,  $i = -r, \dots, s-1$ , where  $y(t)$  is the exact solution of (1.1).

Define  $p$  as in (1.8) and let

$$\begin{aligned}
 D_m(y) &= \begin{pmatrix} h_n p'(t_n + \theta_0 h_n) - h_n f(p(t_n + \theta_0 h_n), t_n + \theta_0 h_n) \\ \dots \\ h_n p'(t_n + \theta_{s-1} h_n) - h_n f(p(t_n + \theta_{s-1} h_n), t_n + \theta_{s-1} h_n) \end{pmatrix} \\
 &= \sum'_{jk} A_{jk} \frac{h_m^{k\tilde{Y}(k)}}{k!} - h_m f\left(\sum'_{jk} B_{jk} \frac{h_m^{k\tilde{Y}(k)}}{k!}, t_m + \theta h_m\right) \quad (3.6)
 \end{aligned}$$

be the amount by which the exact solution fails to satisfy the formula.

Then the formula is defined to be of order k if

$$D_m(y) = O(h^{k+1})$$

or all solutions  $y \in C^{k+1}[t_0, T]$  (this means either that each element of  $E_n(y)$  or equivalently  $\|E_n(y)\|$  is of order  $h^{k+1}$ ). As in Chapter 2, the order of the formula is in general  $\deg p = C-1 = \sum c_i - 1$ , unless the  $\theta_i$  are chosen such that

$$p'(t_n + \theta_i h_n) = y'(t_n + \theta_i h_n) + O(h^C), \quad i = 0, \dots, s-1,$$

in which case the formula is of order  $C$ . To attain the higher order, one calculates the coefficient of  $y_{n+\theta_i h_n}^{(C)}$  in the expansion of  $p'$  about the time  $t_n + \theta_i h_n$  as a function of  $\theta_i$  and solve for  $s$  distinct  $\theta_i$  to make this coefficient zero. This is illustrated in the following example.

Example 3.1  $(-\sqrt{3}/3, \sqrt{3}/3; 2; 2, 2)$

The Hermite basis functions for this method are:

$$\phi_{-1,0}(\tau) = 1/4 \tau^2(\tau - 1)^2 + 3/4 \tau^2(\tau - 1)^2(\tau + 1)$$

$$\phi_{-1,1}(\tau) = 1/4 \tau^2(\tau - 1)^2(\tau + 1)$$

$$\phi_{0,0}(\tau) = (\tau - 1)^2(\tau + 1)^2$$

$$\phi_{0,1}(\tau) = \tau(\tau - 1)^2(\tau + 1)^2$$

$$\phi_{1,0}(\tau) = 1/4 \tau^2(\tau + 1)^2 - 3/4 \tau^2(\tau + 1)^2(\tau - 1)$$

$$\phi_{1,1}(\tau) = 1/4 \tau^2(\tau + 1)^2(\tau - 1)$$

With  $\theta_0 = -\sqrt{3}/3$ ,  $\theta_1 = \sqrt{3}/3$ , we find:

$$\alpha_{0,-1}^0 = -5/6 - 4/9\sqrt{3} \quad \alpha_{0,-1}^1 = -1/9 - 1/18\sqrt{3}$$

$$\alpha_{0,0}^0 = 8/9\sqrt{3} \quad \alpha_{0,0}^1 = -4/9$$

$$\alpha_{0,1}^0 = 5/6 - 4/9\sqrt{3} \quad \alpha_{0,1}^1 = -1/9 + 1/18\sqrt{3}$$

$$\alpha_{1,-1}^0 = -5/6 + 4/9\sqrt{3} \quad \alpha_{1,-1}^1 = -1/9 + 1/18\sqrt{3}$$

$$\alpha_{1,0}^0 = -8/9\sqrt{3} \quad \alpha_{1,0}^1 = -4/9$$

$$\alpha_{1,1}^0 = 5/6 + 4/9\sqrt{3} \quad \alpha_{1,1}^1 = -1/9 - 1/18\sqrt{3}$$

$$\beta_{0,-1}^0 = 5/18 + 1/9\sqrt{3} \quad \beta_{0,-1}^1 = 1/18 + 1/54\sqrt{3}$$

$$\beta_{0,0}^0 = 4/9 \quad \beta_{0,0}^1 = -4/27\sqrt{3}$$

$$\beta_{0,1}^0 = 5/18 - 1/9\sqrt{3} \quad \beta_{0,1}^1 = -1/18 + 1/54\sqrt{3}$$

$$\beta_{1,-1}^0 = 5/18 - 1/9\sqrt{3} \quad \beta_{1,-1}^1 = 1/18 - 1/54\sqrt{3}$$

$$\beta_{1,0}^0 = 4/9 \quad \beta_{1,0}^1 = 4/27\sqrt{3}$$

$$\beta_{1,1}^0 = 5/18 + 1/9\sqrt{3} \quad \beta_{1,1}^1 = -1/18 - 1/54\sqrt{3}$$

The corresponding  $A_{jk}$  and  $B_{jk}$  matrices are:

$$A_{-1,0} = \begin{pmatrix} -5/6 - 4/9\sqrt{3} & 0 \\ -5/6 + 4/9\sqrt{3} & 0 \end{pmatrix} \quad B_{-1,0} = \begin{pmatrix} 5/18 + 1/9\sqrt{3} & 0 \\ 5/18 - 1/9\sqrt{3} & 0 \end{pmatrix}$$

$$A_{-1,1} = \begin{pmatrix} -1/9 - 1/18\sqrt{3} & 0 \\ -1/9 + 1/18\sqrt{3} & 0 \end{pmatrix} \quad B_{-1,1} = \begin{pmatrix} 1/18 + 1/54\sqrt{3} & 0 \\ 1/18 - 1/54\sqrt{3} & 0 \end{pmatrix}$$

$$A_{0,0} = \begin{pmatrix} 5/6 - 4/9\sqrt{3} & 8/9\sqrt{3} \\ 5/6 + 4/9\sqrt{3} & -8/9\sqrt{3} \end{pmatrix} \quad B_{0,0} = \begin{pmatrix} 5/18 - 1/9\sqrt{3} & 4/9 \\ 5/18 + 1/9\sqrt{3} & 4/9 \end{pmatrix}$$

$$A_{0,1} = \begin{pmatrix} -1/9 + 1/18\sqrt{3} & -4/9 \\ -1/9 - 1/18\sqrt{3} & -4/9 \end{pmatrix} \quad B_{0,1} = \begin{pmatrix} -1/18 + 1/54\sqrt{3} & -4/27\sqrt{3} \\ -1/18 - 1/54\sqrt{3} & 4/27\sqrt{3} \end{pmatrix}$$

Note  $\det(A_{0,0}) = -40/27\sqrt{3} \neq 0$  so  $A_{0,0}$  is nonsingular. Because

$$h_n p'(t_n + \theta_i h_n) = h_n y'(t_n + \theta_i h_n) + O(h_n^7)$$

the method is of order 6 (the coefficient of  $h_n^6 y_{n+\theta_i h_n}^{(6)}$  is  $1/30(-\theta_i/9 + (\theta_i)^5) = 0$  when  $\theta_i^2 = 1/3$ ).

Let the past values,  $\tilde{Y}_{m+i}$ ,  $i = -\rho, \dots, -1$ , be exact and choose  $Y_m$  so (3.3) is satisfied. Then the local truncation error is defined to be

$$d_m = Y_m - \tilde{Y}_m.$$

A theorem analogous to Theorem 2.1 holds.

**Theorem 3.1** A formula (3.1) with  $A_{0,0}$  nonsingular and  $\|A_{0,0}^{-1}\|$ ,  $\|A_{0,k}\|$ , and  $\|B_{0,k}\|$  bounded as  $h \rightarrow 0$  is of order  $k$  if and only if  $d_m = O(h^{k+1})$ .

**Proof:** With  $\tilde{Y}_{m+j}$ ,  $Y_m$  defined as above we have

$$D_m = \sum'_{jk} A_{jk} \frac{h_m^{k\tilde{Y}(k)} y_{m+j}^{(k)}}{k!} - h_m f \left( \sum'_{jk} B_{jk} \frac{h_m^{k\tilde{Y}(k)} y_{m+j}^{(k)}}{k!}, t_m + \Theta h_m \right) \quad (3.7)$$

$$\begin{aligned} 0 = & \sum_{j=-\rho}^{-1} \sum_k A_{jk} \frac{h_m^{k\tilde{Y}(k)} y_{m+j}^{(k)}}{k!} \\ & - h_m f \left( \sum_{j=-\rho}^{-1} \sum_k B_{jk} \frac{h_m^{k\tilde{Y}(k)} y_{m+j}^{(k)}}{k!} + \sum_k B_{0,k} \frac{h_m^{kY(k)} y_m^{(k)}}{k!}, t_m + \Theta h_m \right) \\ & + \sum_k A_{0,k} \frac{h_m^{kY(k)} y_m^{(k)}}{k!} \end{aligned} \quad (3.8)$$

Subtracting (3.7) from (3.8) and applying the mean value theorem we see that

$$\sum_k A_{0,k} \frac{h_m^k}{k!} (Y_m^{(k)} - \tilde{Y}_m^{(k)}) + D_m = O(h_m \|Y_m - \tilde{Y}_m\|)$$

(using boundedness of  $B_{0,k}$  and the first  $\tilde{c} - 1$  derivatives of  $f$ ). Thus

$$A_{0,0} d_m = -D_m + O(h_m \|d_m\|). \quad (3.9)$$

Because  $A_{0,0}^{-1}$  exists and is bounded as  $h \rightarrow 0$ ,

$$d_m = -A_{0,0}^{-1} D_m + O(h_m \|d_m\|),$$

and if the method is order  $k$ , then

$$d_m = O(h^{k+1}).$$

Conversely if  $d_m = O(h^{k+1})$ , it follows immediately from (3.9) that

$$D_m = O(h^{k+1}),$$

so the method is of order  $k$ .

Q.E.D.

Lemma 2.1 extends directly to  $\alpha_{ij}^k$  and  $\beta_{ij}^k$  with only a change in notation in the proof (to add the extra index). We restate it as follows:

Lemma 3.1 The coefficients  $\alpha_{ij}^k, \beta_{ij}^k$  are rational functions in the stepsize change variables  $\delta_{n-r+2}, \dots, \delta_n$ .

In particular  $\alpha_{ij}^k, \beta_{ij}^k$  are finite continuous functions of  $\delta_{n-r+2}, \dots, \delta_n$  in a neighborhood of  $(0, \dots, 0)$ . In fact, the rational functions are continuous and finite on the domain

$$\{(\delta_{n-r+2}, \dots, \delta_n) \mid \delta_{n-r+2} > -1, \dots, \delta_n > -1\}$$

as in Chapter 2, since the matrix in (2.11) is nonsingular on this domain.

If we assume that  $h_m$  is chosen by the stepsize selection scheme (2.12)

(with  $n$  replaced by  $m$ ) we get the analog of Corollary 2.1.

Corollary 3.1 If a method described by (3.1) chooses stepsizes by (2.12), then the matrices  $A_{jk}, B_{jk}$  are bounded for all  $h, m$ .

Finally, if  $A_{0,0}^{-1}$  exists when the stepsize is constant, then  $A_{0,0}^{-1}$  exists and is bounded for small stepsize changes, by continuity of the

elements of  $A_{0,0}$ . Thus we have

**Theorem 3.2** Suppose a method described by (3.1) and choosing stepsizes by (2.12) has  $A_{0,0}$  nonsingular when the stepsize is constant and satisfies

$$\delta_i = O(h) \text{ for all } i.$$

Then the method is of order  $k$  if and only if  $d_m = O(h^{k+1})$ .

### 3.2 Stability Regions

The rationale for using block multistep methods is to obtain improved stability regions, since the methods of Chapter 2 were all found to be stiffly stable only for order at most 9. In this section we develop the characteristic polynomial for block multistep methods and exhibit several families, some of which are stiffly stable with order up to 24. The theory in this and the next section can be applied to general methods of the form (3.5) and not just to those methods derived from (3.2).

**Theorem 3.3** A formula of the form (3.5) has a region of absolute stability

$$\{\mu \mid \text{all roots, } \xi, \text{ of } \pi(\mu, \xi) \text{ have } |\xi| \leq 1 \text{ and roots of modulus 1 are simple}\}$$

where

$$\pi(\mu, \xi) = \det \left( \sum_{jk} (A_{jk} \frac{\mu^k}{k!} - B_{jk} \frac{\mu^{k+1}}{k!}) \xi^{\rho+j} \right).$$

**Proof:** To find the absolute stability region of formula (3.5) we apply it to  $y' = \lambda y$  with constant stepsize  $h$  and obtain

$$\sum_{jk} A_{jk} \frac{(h\lambda)^k y_{m+j}}{k!} = h\lambda \left( \sum_{jk} B_{jk} \frac{(h\lambda)^k y_{m+j}}{k!} \right), \quad (3.10)$$

with  $A_{jk}, B_{jk}$  independent of  $h, \lambda, m$ . Writing  $\mu \equiv h\lambda$  and collecting terms, we find that (3.10) takes the form

$$A_{-\rho} Y_{m-\rho} + \dots + A_0 Y_m = 0, \quad (3.11)$$

where

$$A_j = \sum_{k=0}^{\infty} A_{jk} \frac{\mu^k}{k!} - B_{jk} \frac{\mu^{k+1}}{k!} \quad (3.12)$$

is an  $s \times s$  matrix whose elements are polynomials in  $\mu$ . We solve this by converting it to a one step equation.  $A_0(\mu)$  is nonsingular except for finitely many values of  $\mu$ . Except for these values of  $\mu$  we may multiply (3.11) by  $A_0^{-1}$  and obtain

$$Y_m = -A_0^{-1} A_{-1} Y_{m-1} - \dots - A_0^{-1} A_{-\rho} Y_{m-\rho}.$$

If we write

$$Z_m = \begin{pmatrix} Y_{m-1} \\ \dots \\ Y_{m-\rho} \end{pmatrix},$$

then we have

$$Z_{m+1} = AZ_m$$

where

$$A = \begin{pmatrix} -A_0^{-1} A_{-1} & -A_0^{-1} A_{-2} & \dots & -A_0^{-1} A_{-\rho+1} & -A_0^{-1} A_{-\rho} \\ I & 0 & \dots & 0 & 0 \\ 0 & I & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & I & 0 \end{pmatrix} \quad (3.13)$$

Thus the solution is

$$Z_m = A^m Z_0,$$

where  $Z_0$  must be defined by some other means.

By writing  $A^m = P^{-1} J^m P$  where  $J$  is the Jordan canonical form of  $A$ , we can see that the elements of  $A^m$  are linear combinations of

$$\{\xi_i^m, m\xi_i^m, \dots, m^r \xi_i^m \mid \xi_i \text{ is an eigenvalue}$$

of A with Jordan block of size r\},

with coefficients independent of m. Thus for  $Z_m$  to remain bounded independently of the choice of  $Z_0$ , all eigenvalues of A must be at most 1 in magnitude, and those eigenvalues of magnitude 1 must correspond to simple Jordan blocks. To establish the theorem it is only necessary to calculate  $\det(\xi I - A)$ . Since A is a companion matrix we see that

$$\det(\xi I - A) = \det(\xi^\rho + A_0^{-1}A_{-1}\xi^{\rho-1} + \dots + A_0^{-1}A_{-\rho}) .$$

This polynomial has the same roots if we multiply it by  $\det(A_0)$ , so the eigenvalues are the solutions to the equation

$$\det(A_0\xi^\rho + A_{-1}\xi^{\rho-1} + \dots + A_{-\rho}) = 0 .$$

The left side is the polynomial denoted by  $\pi$  in the statement of the theorem.

Finally note that the values of  $\mu$  which make  $A_0$  singular result in an infinite root  $\xi$  and hence occur outside the region of absolute stability. For if  $A_0(\mu)$  is singular then a similarity transform (which preserves determinants) may be applied to  $A_0\xi^\rho + \dots + A_{-\rho}$  to replace  $A_0$  by a matrix with zero first column. This results in a stability polynomial of at most degree  $\rho-1$  in  $\xi$  which thus has at least one root at  $\infty$  (in the sense of algebraic functions). Thus the use of  $A_0^{-1}$  is justified. Q.E.D.

Example 3.2 BLOCK3:  $(1/2, 3/2, 5/2; 1^r; 2, 2, 2) \quad r = 1, \dots, 18$

This family starts at order 6 and avoids the problem that many families have of starting at very low order and having to take many small steps initially. The methods are almost A-stable up to order 15 (i.e.,  $\alpha > 89^\circ$ ), which is probably sufficient for most applications. The

stability regions then worsen slowly through order 20, up to which  $\alpha$  is still at least  $80^\circ$ . The results of Section 3.1 show that the order of  $(1/2, 3/2, 5/2:1^r; 2,2,2)$  is  $r + 5$ . The stability regions illustrated in Figs. 3.1-3.3 show that the methods are stiffly stable for  $r \leq 18$ . The corresponding D's and  $\alpha$ 's are given in Table 3.1. Note that by taking three future points the stability regions are dramatically improved over the stability regions of methods with one future point.

Table 3.1

BLOCK3  $(1/2, 3/2, 5/2:1^r; 2,2,2)$ 

r	order	D	$\alpha(^{\circ})$
1	6	-0.369	78.6
2	7	-0.118	86.6
3	8	-0.022	89.3
4	9	-0.001	89.9
5	10	0	90
6	11	0	90
7	12	0	90
8	13	0	90
9	14	-0.007	89.8
10	15	-0.025	89.4
11	16	-0.058	88.7
12	17	-0.106	87.7
13	18	-0.172	86.3
14	19	-0.264	84.4
15	20	-0.401	81.5
16	21	-0.710	76.6
17	22	-1.397	66.3
18	23	-7.515	39.4

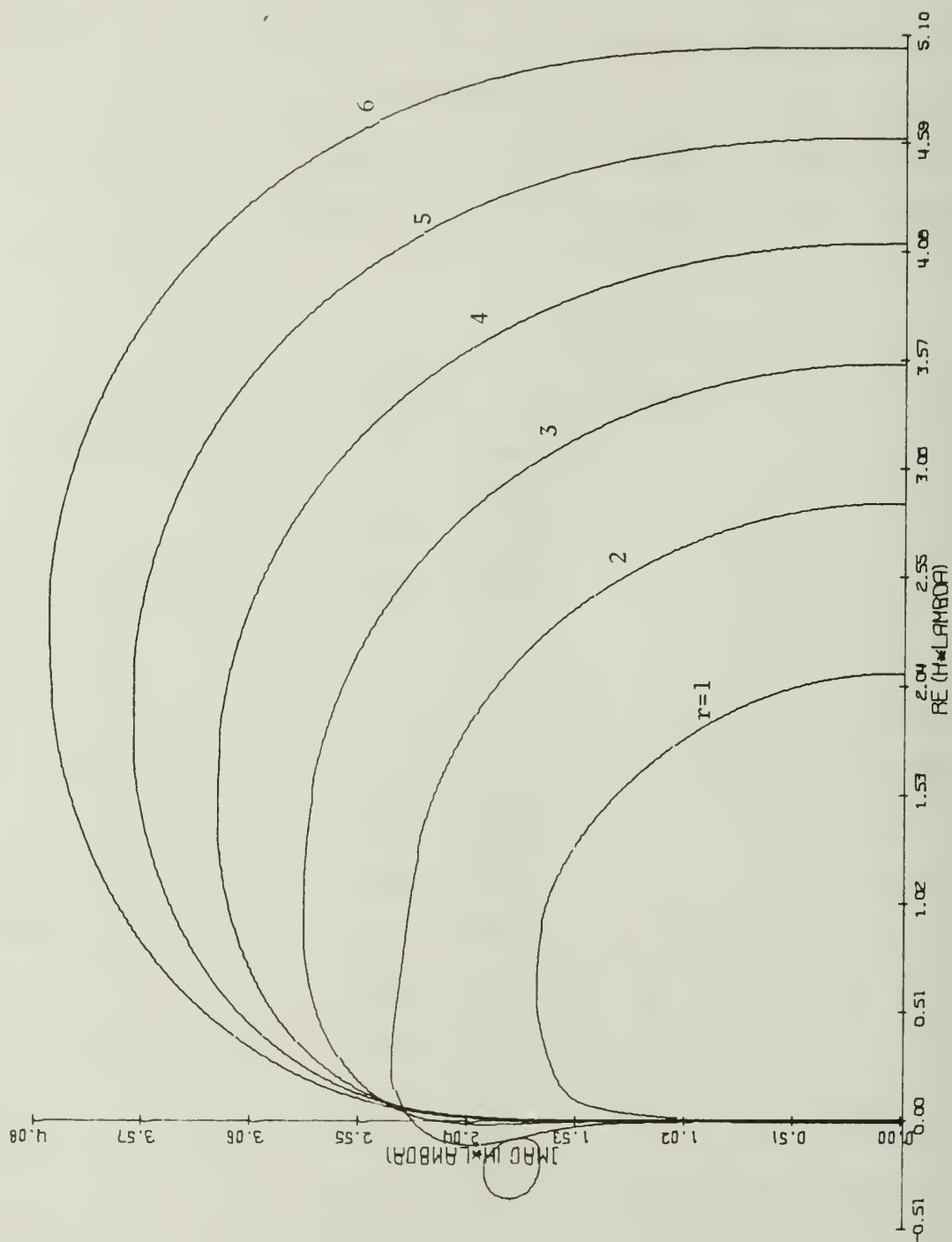


Fig. 3.1. BLOCK3  $(1/2, 3/2, 5/2; 1^r; 2, 2, 2, 2)$ ,  $r = 1, \dots, 6$  Order 6-11

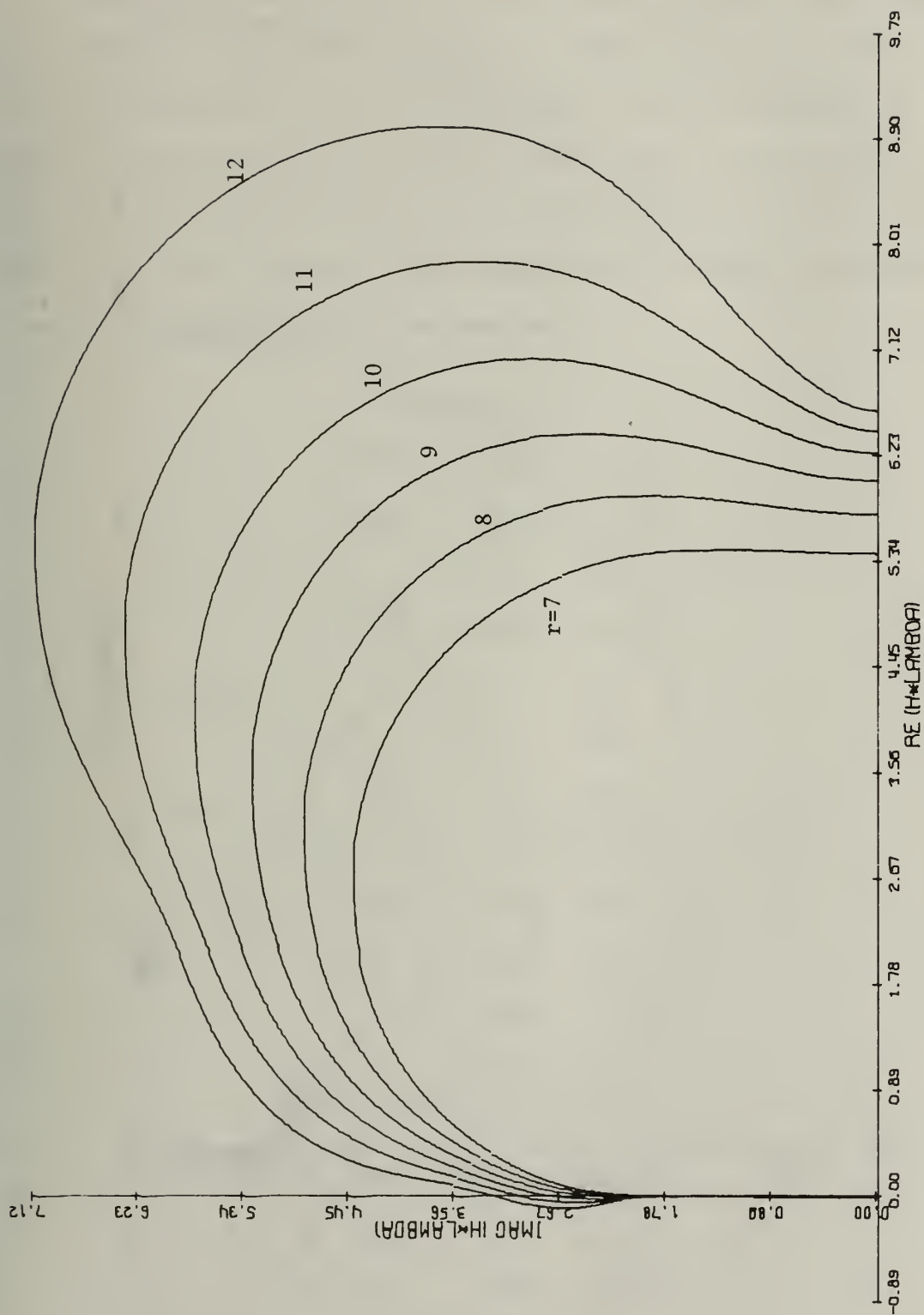


Fig. 3.2. BLOCK3  $(1/2, 3/2, 5/2; 1^r; 2, 2, 2), r = 7, \dots, 12$  Order 12-17

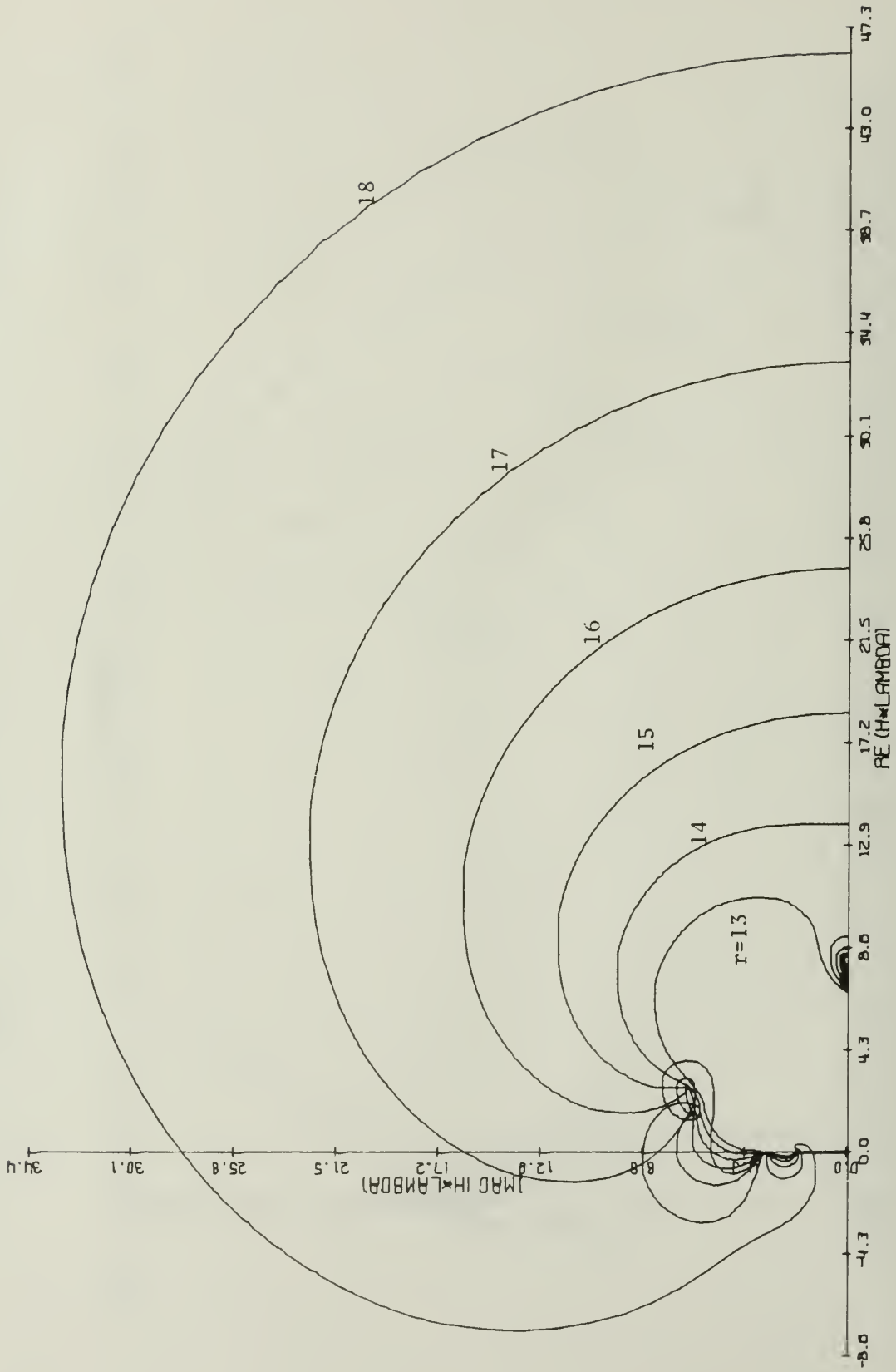


Fig. 3.3. BLOCK3  $(1/2, 3/2, 5/2; 1^r; 2, 2, 2), r = 13, \dots, 18$  Order 18-23

Example 3.3 BLOCK2:  $(1/2, 3/2:1^r; 2, 2)$   $r = 1, \dots, 14$

This family is stiffly stable up to order 17 and almost A-stable up to order 9. The stability regions are shown in Figs. 3.4 and 3.5 with the corresponding D's and  $\alpha$ 's in Table 3.2. Note that although the stability regions of these methods are much better than the methods in Chapter 2, they are distinctly inferior to those of Example 3.2. Thus if high accuracy were desired in the integration of the ODE, the methods of Example 3.2 would probably be preferable.

Table 3.2

BLOCK2  $(1/2, 3/2:1^r; 2, 2)$

r	order	D	$\alpha(^{\circ})$
1	4	-0.101	85.7
2	5	-0.004	89.8
3	6	0	90
4	7	0	90
5	8	0	90
6	9	-0.007	89.8
7	10	-0.058	88.6
8	11	-0.180	86.2
9	12	-0.391	82.6
10	13	-0.715	77.8
11	14	-1.192	71.9
12	15	-1.921	64.8
13	16	-3.164	56.9
14	17	-5.620	47.7

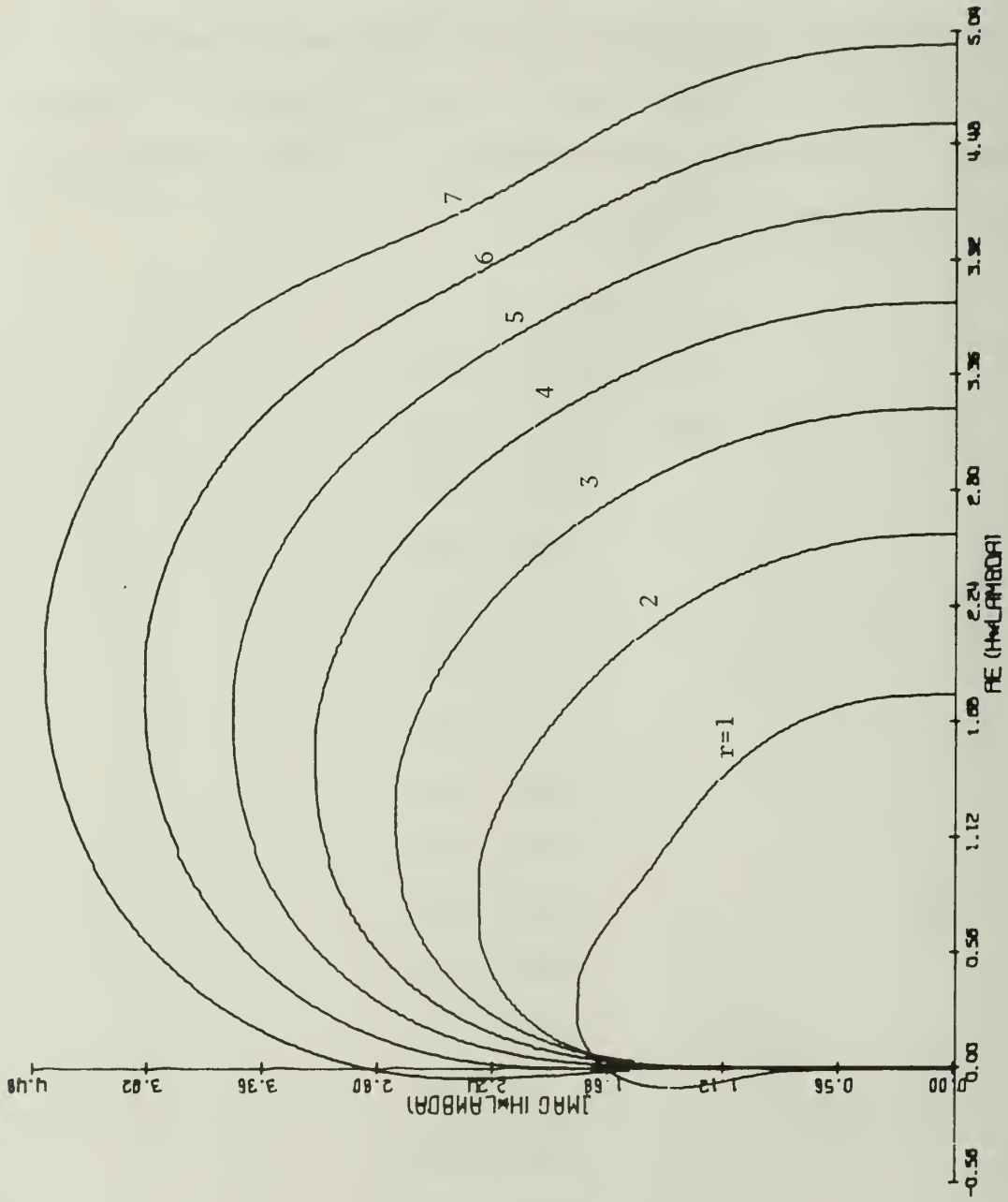


Fig. 3.4. BLOCK2  $(1/2, 3/2; 1^r; 2, 2), r = 1, \dots, 7$  Order 4-10

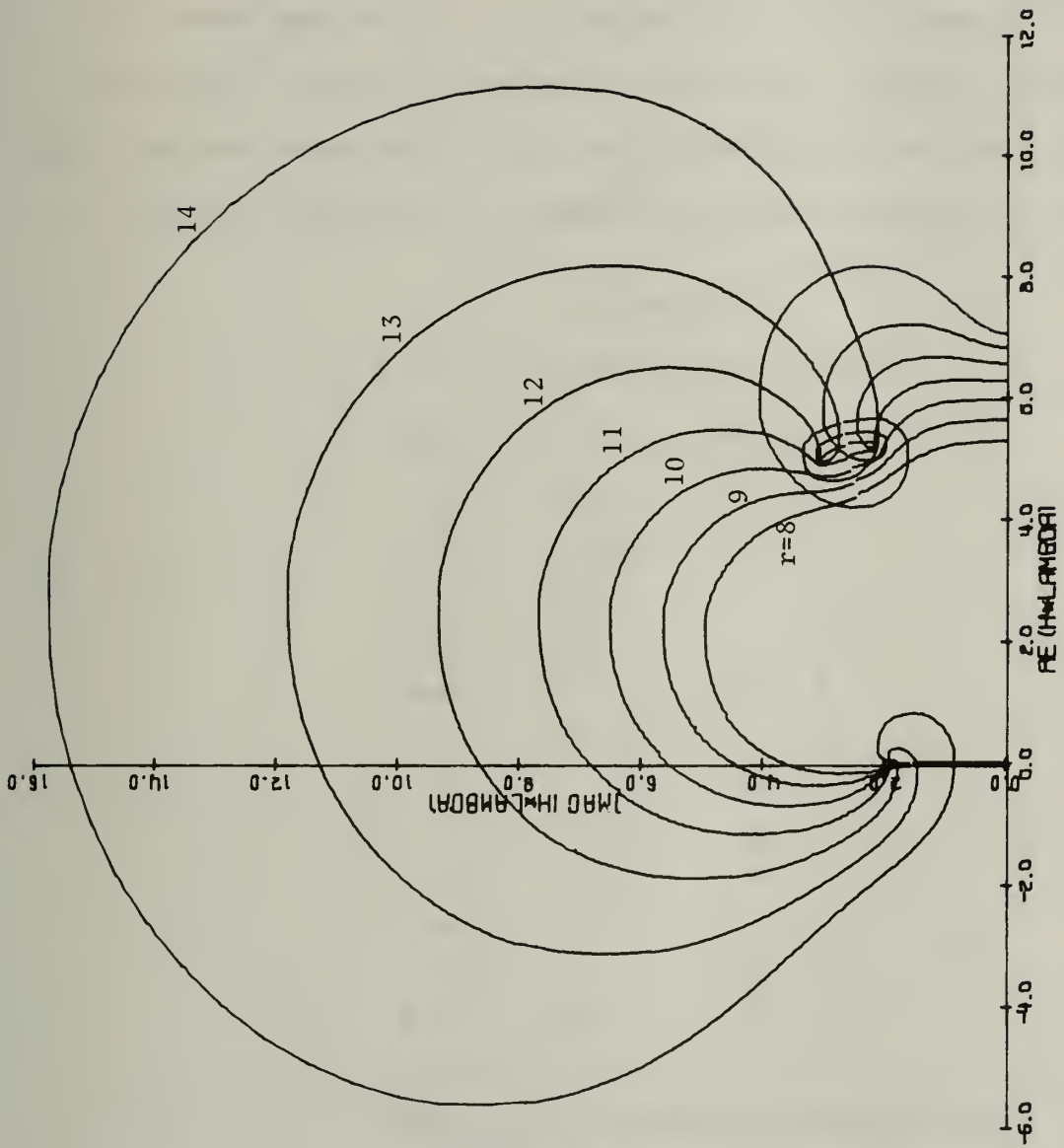


Fig. 3.5. BLOCK2  $(1/2, 3/2; 1; 2, 2)$ ,  $r = 8, \dots, 14$  Order 11-17

Example 3.4 BLOCK4:  $(1/2, 3/2, 5/2, 7/2:1^r; 2, 2, 2, 2)$   $r = 1, \dots, 17$

This family is stiffly stable up to order 24 and almost A-stable up to order 20. The stability regions are shown in Figs. 3.6-3.8, with the corresponding D's and  $\alpha$ 's in Table 3.3. For very high order the stability regions are superior to those of Example 3.2, but the improvement is probably not enough to justify the computer time required to solve the implicit equations for one more future point. Thus unless extremely high accuracy is desired the methods of Example 3.2 seem more practical.

Table 3.3

BLOCK4  $(1/2, 3/2, 5/2, 7/2:1^r; 2, 2, 2, 2)$

r	order	D	$\alpha(^{\circ})$
1	8	-0.705	71.1
2	9	-0.358	81.2
3	10	-0.162	86.2
4	11	-0.059	88.6
5	12	-0.014	89.6
6	13	-0.001	90.0
7	14	0	90
8	15	0	90
9	16	0	90
10	17	0	90
11	18	0	90
12	19	-0.011	89.8
13	20	-0.047	89.1
14	21	-0.873	83.4
15	22	-2.581	72.7
16	23	-5.565	56.6
17	24	-10.993	25.8

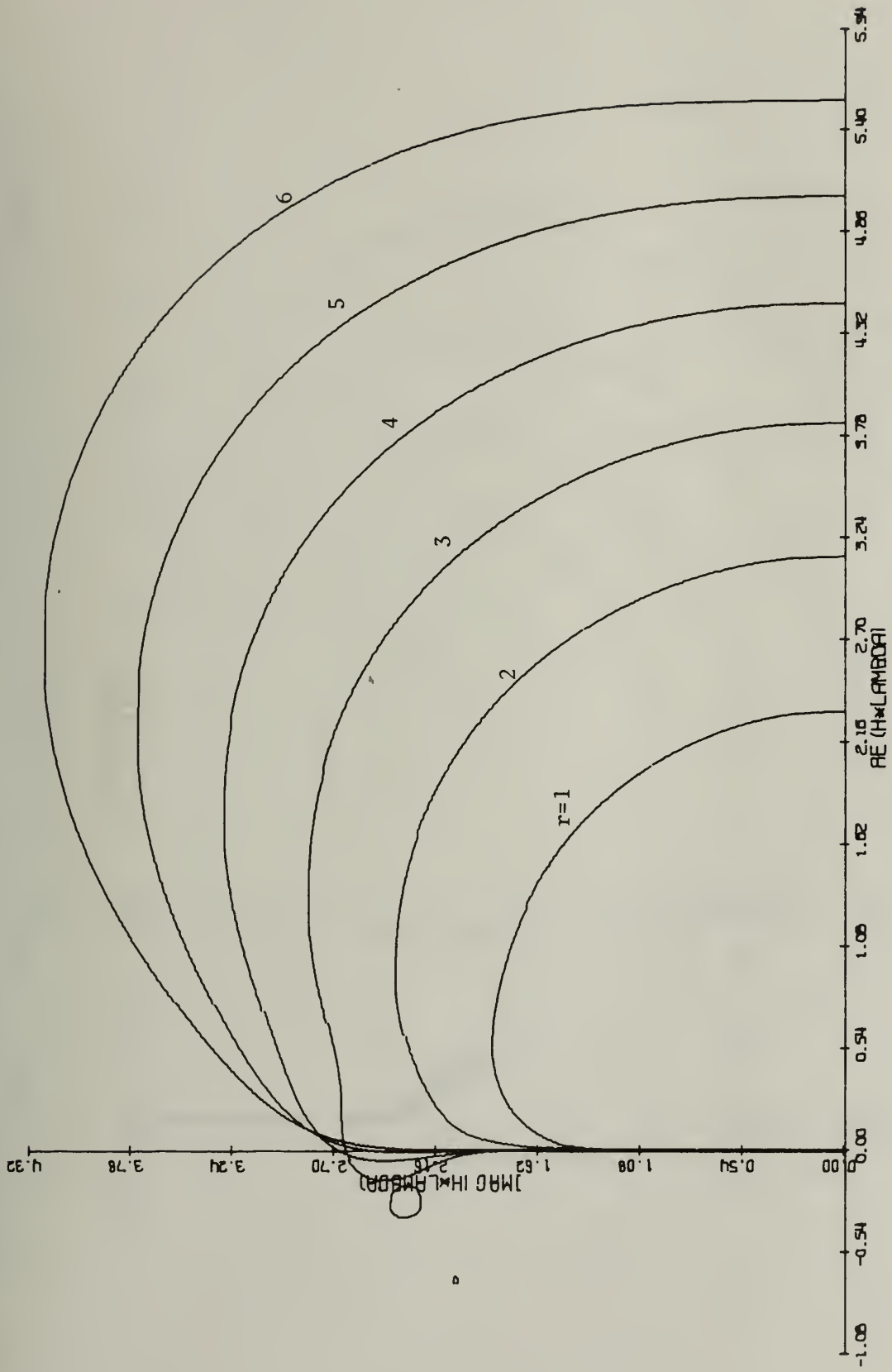


Fig. 3.6. BLOCK4  $(1/2, 3/2, 5/2, 7/2; 1^r; 2, 2, 2, 2), r = 1, \dots, 6$  Order 8-13

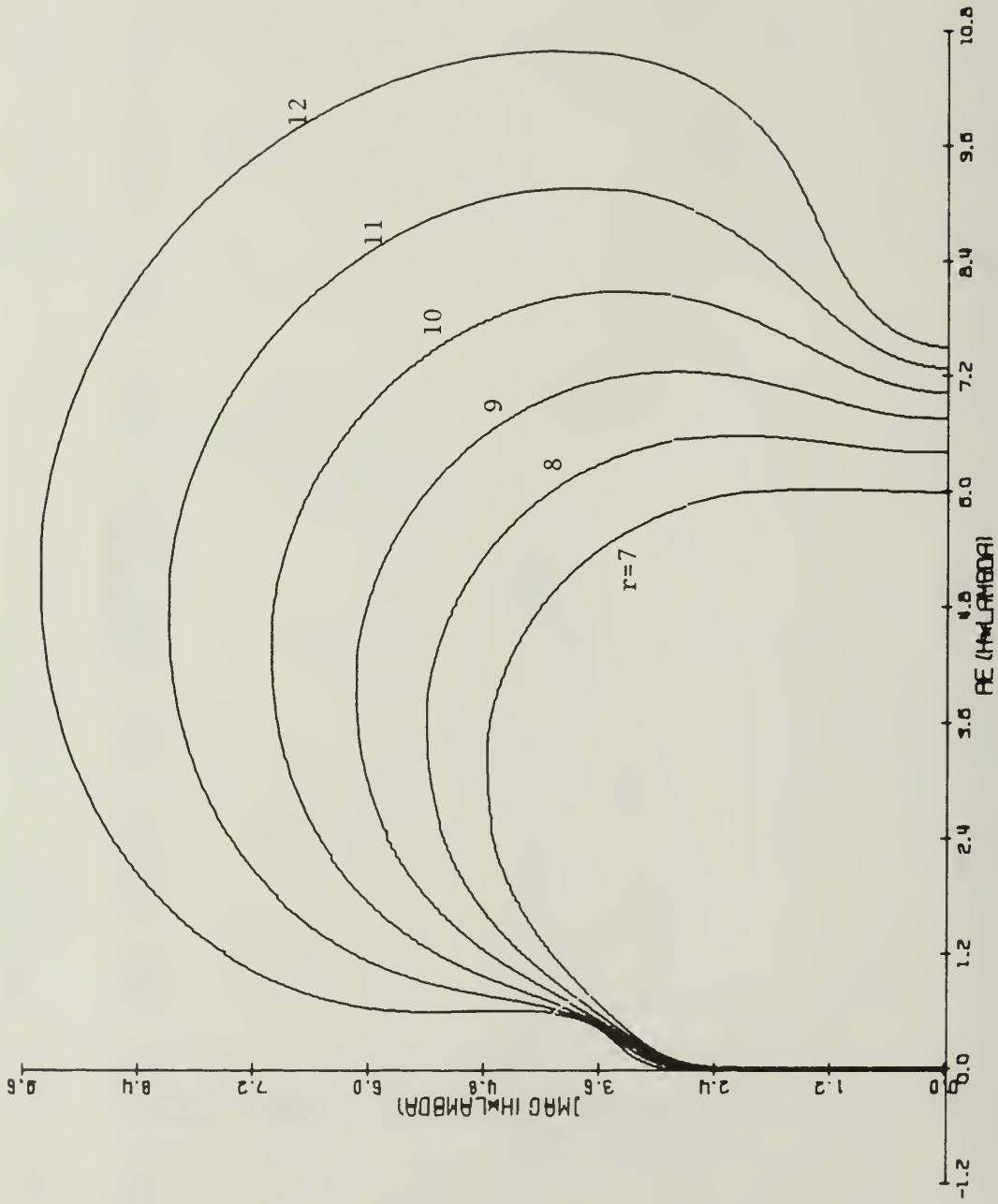


Fig. 3.7. BLOCK4  $(1/2, 3/2, 5/2, 7/2; 1^r; 2, 2, 2, 2), r = 7, \dots, 12$  Order 14-19

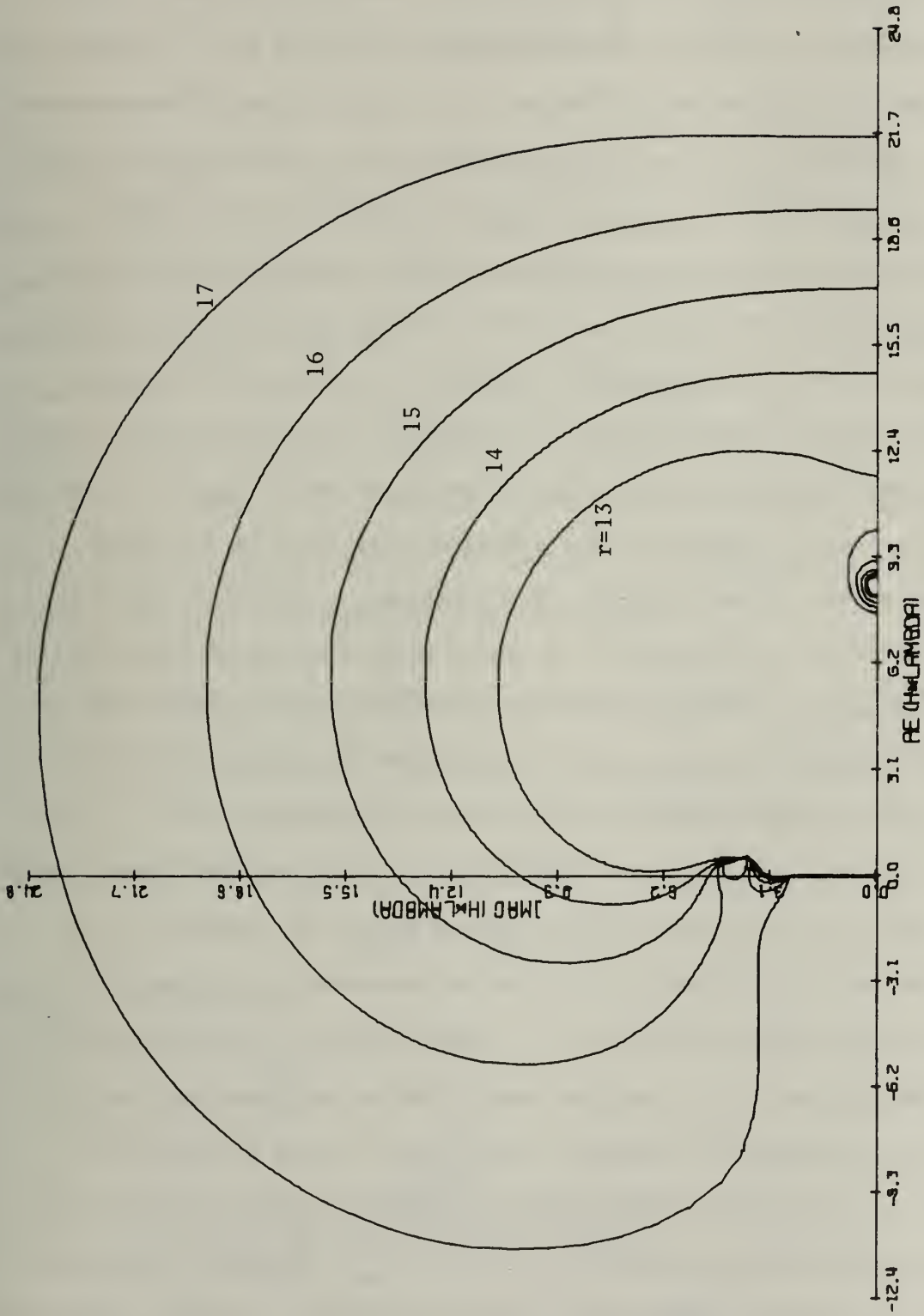


Fig. 3.8. BLOCK4  $(1/2, 3/2, 5/2, 7/2; 1^r; 2, 2, 2, 2), r = 13, \dots, 17$  Order 20-24

### 3.3 Stability and Convergence

In this section we obtain sufficient conditions for the stability and convergence of block multistep methods of the form (3.1). These conditions apply to the methods of Chapter 2 as a special case. The results in this section are an extension of results of Gear and Watanabe (1974).

Because stability and convergence are properties of a method rather than of the underlying formulas, we must consider the technique used to change stepsizes and the schemes used to select stepsizes and formulas, as well as the formulas themselves. Of these, the stepsize changing technique needs the most explanation. As mentioned above, either the variable step or the interpolation technique may be used. The variable step technique is simpler to handle theoretically because the saved derivatives are defined in terms of the values  $y_{n+i}$  by (1.4). Thus to show stability of the method, it is sufficient to establish the stability of the values  $y_{n+i}$ , assuming  $f$  and its derivatives are bounded. For this reason, the stability theory will consider only the values  $y_{n+i}$  to be saved, even though a code would also save the derivatives.

With the interpolation technique it is no longer true after a stepsize change that the saved derivatives are defined by equations (1.4), since the new derivatives are those of an interpolation polynomial at noncollocating points. Thus we must examine the error amplification for these derivatives. For notational simplicity we will restrict  $\tilde{c}$  to be 2 with the interpolation technique. Furthermore, as was discussed in Section 2.3, the interpolation technique depends on the saved data used in the interpolating polynomial. Let  $(R + s)_{\max} = \max\{(\rho+1)s\}$  where the maximum is taken over all formulas used in the method. We will assume for simplicity that the saved data before step  $m$  is

$$(y_{n-1}, \dots, y_{n-(R+s)_{\max}}, h_n y'_{n-1}, \dots, h_n y'_{n-(R+s)_{\max}})^T.$$

Fewer values and derivatives could be used in defining the interpolating polynomial in special cases. For example let  $r_m^0, r_m^1$  be integers such that the formula used at the  $m^{\text{th}}$  step satisfies  $-r_m^0 \leq -r$  and  $c_i = 1$  for  $i < -r_m^1$ . Then the values  $y_{n+i}$  for  $-r_m^0 \leq i \leq s-1$  and  $hy'_{n+i}$  for  $-r_m^1 \leq i \leq s-1$  could be used to define the interpolation polynomial for stepsize changing. The formula selection scheme and  $r_m^0, r_m^1$  must be chosen to ensure that the past data used by the next formula are correctly spaced. For example in the method implemented in Appendix I,  $r_m^0 = r, r_m^1 = 0$  and the formula changing scheme allows  $r$  to increase by at most one after each step. This abbreviated data structure would complicate the notation used in this section, but would not effect the validity of the stability and convergence results. Note that the polynomial used in the interpolation technique is in general distinct from the polynomial defining the formula.

We first examine the effect of one step of formula (3.5), with variable step technique, on the error. Let  $Y_{m+1}, \tilde{Y}_{m+1}$  be the numerical and exact solution, respectively, grouped as  $s$ -vectors (c.f. (3.4)). Denote the error and its formal derivatives by

$$E_{m+j}^{(k)} = Y_{m+j}^{(k)} - \tilde{Y}_{m+j}^{(k)}.$$

Then by (3.5) and (3.6)

$$\begin{aligned} 0 = & \sum_{jk}' A_{jk} \frac{h_m^k E_{m+j}^{(k)}}{k!} \\ & - h_m \left[ f \left( \sum_{jk}' B_{jk} \frac{h_m^k Y_{m+j}^{(k)}}{k!}, t_m + \Theta h_m \right) \right. \\ & \left. - f \left( \sum_{jk}' B_{jk} \frac{h_m^k \tilde{Y}_{m+j}^{(k)}}{k!}, t_m + \Theta h_m \right) \right] \\ & + D_m. \end{aligned}$$

Let  $J_m$  denote a matrix whose  $i^{\text{th}}$  row is

$$\left( \frac{\partial f_i}{\partial y_j} (\xi_i) \right)_j.$$

Then we choose the  $\xi_i$  such that

$$0 = \sum'_{jk} A_{jk} \frac{h_m^{kE(k)} m^{m+j}}{k!} - h_m^J \sum'_{jk} B_{jk} \frac{h_m^{kE(k)} m^{m+j}}{k!} + D_m.$$

Similarly we can find linear operators  $K_{m,j}^k$  with norm bounded independently of  $m, j$  (assuming that the  $(k+1)^{\text{st}}$  partials of  $f$  are bounded) such that

$$0 = \left( \sum'_{jk} A_{jk} \frac{h_m^{kK^k} m^{mj}}{k!} - h_m^J \sum'_{jk} B_{jk} \frac{h_m^{kK^k} m^{mj}}{k!} \right) E_{m+j} + D_m.$$

Define

$$\varepsilon_m = (E_m, \dots, E_{m-\rho+1})^T,$$

$$d_m = (C_{m,0}^{-1} D_m, 0, \dots, 0)^T.$$

Then we can write

$$\varepsilon_m = S_m \varepsilon_{m-1} - d_m$$

where

$$S_m = \begin{pmatrix} -C_{m,0}^{-1} C_{m,-1} & -C_{m,0}^{-1} C_{m,-2} & \dots & -C_{m,0}^{-1} C_{m,-\rho} \\ I & 0 & \dots & 0 \\ 0 & I & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 \end{pmatrix} \quad (3.14)$$

with

$$C_{m,j} = \sum_k A_{jk} \frac{h_m^{kK^k} m^{mj}}{k!} - h_m^J \sum_k B_{jk} \frac{h_m^{kK^k} m^{mj}}{k!},$$

provided  $C_{m,0}$  is invertible. If the stepsizes are chosen according to

the standard stepsize scheme (2.12) with  $\delta_i = O(h)$ , then  $A_{jk}$  and  $B_{jk}$  are bounded, so  $C_{m,0} = A_{0,0} + O(h)$ . Thus for  $h$  small (which we will assume in stability or convergence proofs), nonsingularity of  $C_{m,0}$  is equivalent to nonsingularity of  $A_{0,0}$  which is in turn equivalent to nonsingularity of  $A_{0,0}$  for the fixed step formula as was discussed in Section 3.1.

Let

$$S_m = \bar{S}_m + h_m \tilde{S}_m$$

where

$$\bar{S}_m = \begin{pmatrix} -A_{0,0}^{-1}A_{-1,0} & -A_{0,0}^{-1}A_{-2,0} & \cdots & -A_{0,0}^{-1}A_{-\rho,0} \\ I & 0 & \cdots & 0 \\ 0 & I & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}. \quad (3.15)$$

Thus

$$\tilde{S}_m = \frac{1}{h_m} \begin{pmatrix} A_{0,0}^{-1}A_{-1,0} - C_{m,0}^{-1}C_{m,-1} & \cdots & A_{0,0}^{-1}A_{-\rho,0} - C_{m,0}^{-1}C_{m,-\rho} \\ 0 & \cdots & 0 \\ \cdots & \cdots & \cdots \\ 0 & \cdots & 0 \end{pmatrix}.$$

We wish to show  $\tilde{S}_m = O(1)$ , as  $h \rightarrow 0$ . Consider one of the matrices making up the first row:

$$\begin{aligned} \frac{1}{h_m} (A_{0,0}^{-1}A_{j,0} - C_{m,0}^{-1}C_{m,j}) = \\ C_{m,0}^{-1}A_{0,0}^{-1} \left( \frac{C_{m,0}A_{j,0} - A_{0,0}C_{m,j}}{h_m} \right) + \left( \frac{A_{0,0}^{-1}C_{m,0}^{-1} - C_{m,0}^{-1}A_{0,0}^{-1}}{h_m} \right) C_{m,0}A_{j,0}. \end{aligned}$$

The first quantity in brackets is clearly bounded as  $h_m \rightarrow 0$ , since every term in the numerator has a positive power of  $h_m$ . The second quantity is

also bounded as  $h_m \rightarrow 0$  as a result of the following two lemmas. Thus  $\tilde{S}_m = O(1)$  as claimed.

**Lemma 3.2** If  $C_m = A_m - h_m B_m$  with  $A_m, B_m$  bounded,  $A_m$  nonsingular and  $A_m^{-1}$  bounded, then for small enough  $h_m$ ,  $C_m^{-1} = A_m^{-1} - h_m D_m$  with  $D_m$  bounded.

**Proof:**

$$\begin{aligned} C_m^{-1} &= [A_m (I - h_m A_m^{-1} B_m)]^{-1} = \sum_{i=0}^{\infty} (h_m A_m^{-1} B_m)^i A_m^{-1} \\ &= A_m^{-1} + h_m \sum_{i=1}^{\infty} h_m^{i-1} (A_m^{-1} B_m)^i A_m^{-1} \\ &= A_m^{-1} + h_m \sum_{i=0}^{\infty} h_m^i (A_m^{-1} B_m)^i \\ &= A_m^{-1} + h_m [I - h_m A_m^{-1} B_m]^{-1} A_m^{-1} B_m A_m^{-1}. \end{aligned}$$

Thus we may let  $D_m = [I - h_m A_m^{-1} B_m]^{-1} A_m^{-1} B_m A_m^{-1}$ . Q.E.D.

**Lemma 3.3** If  $C_m = A_m + h_m B_m$  with  $A_m, B_m$  bounded, then

$$A_m C_m - C_m A_m = h_m D_m$$

with  $D_m$  bounded.

The proof is a trivial calculation.

Finally we note that by Lemma 3.1 we can write

$$\overline{S}_m = \hat{S}_m + h_m \overline{\overline{S}}_m$$

where  $\hat{S}_m$  is the matrix (3.15) for the constant stepsize method and  $\overline{\overline{S}}_m$  is bounded. Thus we have the following

**Lemma 3.4** Suppose a method described by (3.1) and choosing stepsizes by (2.12) with  $\delta_i = O(h)$  has  $A_{0,0}$  nonsingular when the stepsize is constant.

Suppose the method is applied to solve equation (1.1) where  $f$  has  $\tilde{c} - 1$  continuous partial derivatives. Then the error at the  $m^{\text{th}}$  step satisfies

$$\epsilon_m = S_m \epsilon_{m-1} - d_m$$

where

$$S_m = \hat{S}_m + O(h) .$$

We next consider the effect of one step of the interpolation technique. As mentioned above,  $\tilde{c}$  is restricted to 2. To include the effects of the method on the derivatives, we will extend the  $\epsilon_m$  vectors to include derivatives. Because formulas using different  $s$  may occur in a method, all the saved values  $y_{n+i}$  will be grouped together. It is convenient first to calculate a permutation of the error vector which groups the values and derivatives together in blocks of length  $2s$ . As we shall see, the interpolation process introduces an extra past block into the error equation. To make room for this block, the error vectors will be further extended and the error amplification matrices  $S_m$  will have columns of zeroes added.

Let the points  $t_{n+i}$  be spaced equally with stepsize  $h_m$ , and let

$$Z_{m+j} = (Y_{m+j}, h_m Y'_{m+j})^T ,$$

$$\tilde{Z}_{m+j} = (\tilde{Y}_{m+j}, h_m \tilde{Y}'_{m+j})^T ,$$

$$A_j = \begin{pmatrix} A_{j,0} & A_{j,1} \\ 0 & \delta_{j,0} I \end{pmatrix} ,$$

$$B_j = \begin{pmatrix} B_{j,0} & B_{j,1} \\ \delta_{j,0} I & 0 \end{pmatrix} ,$$

and define

$$H_{m+j} = Z_{m+j} - \tilde{Z}_{m+j}.$$

Then (3.5) and the equations  $y'_{n+js+i} = f(y_{n+js+i}, t_{n+js+i})$  become

$$\sum_{j=-\rho}^0 A_j Z_{m+j} = h_m f\left(\sum_{j=-\rho}^0 B_j Z_{m+j}, t_m + \Theta h_m\right), \quad (3.16)$$

where now

$$\Theta = (\theta_0, \dots, \theta_{s-1}, s-1, \dots, 0)^T.$$

If we extend  $D_m$  to be a  $2s$ -vector by adding zeroes at the end, equations (3.16) and (3.6) yield

$$\begin{aligned} 0 = & \sum_{j=-\rho}^0 A_j H_{m+j} \\ & - h_m \left( f\left(\sum_{j=-\rho}^0 B_j Z_{m+j}, t_m + \Theta h_m\right) \right. \\ & \quad \left. - f\left(\sum_{j=-\rho}^0 B_j \tilde{Z}_{m+j}, t_m + \Theta h_m\right) \right) \\ & + D_m. \end{aligned}$$

As above, we may choose  $\bar{J}_m$  bounded independently of  $h, m$  such that

$$0 = \sum_{j=-\rho}^0 A_j H_{m+j} - h_m \bar{J}_m \sum_{j=-\rho}^0 B_j H_{m+j} + D_m.$$

Let  $C_{m,j} = A_j - h_m \bar{J}_m B_j$ . If we define

$$\bar{e}_m = (H_m, \dots, H_{m-\rho})^T,$$

$$\bar{d}_m = (-C_{m,0}^{-1} D_m, 0, \dots, 0)^T,$$

then we can write

$$\bar{e}_m = \bar{S}_m \bar{e}_{m-1}^m - d_m, \quad (3.17)$$

where

$$\bar{S}_m = \begin{pmatrix} -C_{m,0}^{-1}C_{m,-1} & \cdots & -C_{m,0}^{-1}C_{m,-\rho} & 0 \\ I & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & I & 0 \end{pmatrix}.$$

The superscript on  $\bar{\epsilon}_{m-1}^m$  indicates that the data points are spaced with stepsize  $h_m$  instead of the standard spacing corresponding to the  $(m-1)^{st}$  step,  $h_{m-1}$ . Note  $C_{m,0}$  is invertible for small  $h_m$  because

$$A_0^{-1} = \begin{pmatrix} A_{0,0}^{-1} & -A_{0,0}^{-1}A_{0,1} \\ 0 & I \end{pmatrix}.$$

As before,

$$\bar{S}_m = \hat{S}_m + O(h)$$

where

$$\hat{S}_m = \begin{pmatrix} -A_0^{-1}A_{-1} & \cdots & -A_0^{-1}A_{-\rho} & 0 \\ I & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & I & 0 \end{pmatrix}.$$

To allow formula changes we permute the elements of the error vector  $\bar{\epsilon}_m$  and define

$$\epsilon_m = (E_m, \dots, E_{m-\rho}, h_m E'_m, \dots, h_m E'_{m-\rho})^T.$$

Then (3.17) becomes

$$\epsilon_m = S_m \epsilon_{m-1}^m - d_m,$$

where  $S_m$  corresponds to  $\bar{S}_m$  and  $d_m$  is the permuted version of  $\bar{d}_m$ . We have

$$S_m = \hat{S}_m + O(h)$$

where

$$\hat{S}_m = \left[ \begin{array}{cccc|cccc} -A_{0,0}A_{-1,0} & \dots & -A_{0,0}A_{-\rho,0} & 0 & -A_{0,0}A_{-1,1} & \dots & -A_{0,0}A_{-\rho,1} & 0 \\ I & \dots & 0 & 0 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & I & 0 & 0 & \dots & 0 & 0 \\ \hline 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & \dots & 0 & 0 & I & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & 0 & 0 & \dots & I & 0 \end{array} \right]. \quad (3.18)$$

With the interpolation technique, the saved data is preprocessed to change the stepsize before the constant stepsize method is applied. We now add the effect of this preprocessing to the error equation. At the  $(m-1)^{st}$  step before the change in stepsize we have saved data

$$(Y_{m-1}, \dots, Y_{m-\rho-1}, h_{m-1}Y'_{m-1}, \dots, h_{m-1}Y'_{m-\rho-1})^T$$

whose values correspond to  $t_{m-1}^{old} + ih_{m-1}$ . Denote this vector by  $T_{m-1}$ , and

the corresponding exact values by  $\tilde{T}_{m-1}$ . Then  $\epsilon_{m-1}^{old} = T_{m-1} - \tilde{T}_{m-1}$  is the

error before the stepsize is changed. The polynomial which interpolates the  $2R$  values and derivatives of  $T_{m-1}^{old}$  is evaluated at the new time steps

$t_{m+i} = t_{m-1}^{old} + (s-1)h_{m-1} + h_m i$  to give the past values after the stepsize

change (denoted by  $T_{m-1}^m$  with corresponding exact values  $\tilde{T}_{m-1}^m$ ). This

evaluation is a linear transformation which may be written

$$C_m = Q^{-1}C(h_m/h_{m-1})Q$$

where  $Q$  changes the saved data into Nordsieck form at time  $t_{m-1} + (s-1)h_{m-1}$ , and  $C(\alpha) = \text{diag}(1, \alpha, \dots, \alpha^{2R-1})$ .

Thus

$$\tau_{m-1}^m = C_m \tau_{m-1}^m ,$$

$$\tilde{\tau}_{m-1}^m = C_m \tilde{\tau}_{m-1}^m + O(h^{2R}) ,$$

$$\epsilon_{m-1}^m = C_m \epsilon_{m-1}^m + O(h^{2R}) .$$

Since  $2R \geq C$  we can modify the discretization term  $d_m$  to include the  $O(h^{2R})$  term above. Then the process of stepsize changing and application of the formula can be combined to give the error equation

$$\epsilon_m = S_m C_m \epsilon_{m-1} - d_m .$$

If the stepsize changing scheme (2.12) is used with  $\delta_i = O(h)$  for all  $i$ , the  $h_m/h_{m-1} = 1 + O(h)$ , so  $C_m = I + O(h)$ . Thus  $S_m C_m = \hat{S}_m + O(h)$  and Lemma 3.4 holds for the interpolation technique (with the symbols suitably reinterpreted, and replacing  $S_m$  by  $S_m C_m$ ).

Lemma 3.4 gives the effect of one step using either the variable step or interpolation technique on a fixed formula (3.1). Changing formulas can be allowed by extending the error vectors to include the greatest number of past points allowed, and extending the error amplification matrices suitably. If the variable step technique is used then the  $\epsilon_m$  vectors are extended to save  $R_{\max}$  values and the error amplification matrices are extended to the  $R_{\max} \times R_{\max}$  matrices

$$\begin{pmatrix} S_m & 0 \\ L & D \end{pmatrix}$$

where  $L$  and  $D$  are chosen so that the extended  $S$  matrix has ones on the  $s^{\text{th}}$  subdiagonal. With the interpolation technique the  $\epsilon_m$  vectors are extended to save  $(R + s)_{\max}$  values and derivatives. If the error amplification matrix is partitioned into  $(R + s) \times (R + s)$  blocks,

$$S_m = \begin{pmatrix} S_{0,0} & S_{0,1} \\ S_{1,0} & S_{1,1} \end{pmatrix},$$

then  $S_m$  is extended to the  $2(R + s)_{\max} \times 2(R + s)_{\max}$  matrix

$$\begin{pmatrix} S_{0,0} & 0 & S_{0,1} & 0 \\ L & D & 0 & 0 \\ S_{1,0} & 0 & S_{1,1} & 0 \\ 0 & 0 & L & D \end{pmatrix}$$

where  $L$  and  $D$  are as above. We assume this to be done with no change in notation. In addition, the interpolation technique is assumed to change all  $2(R + s)_{\max}$  saved values and derivatives to the new stepsize, which redefines  $C_m$  appropriately.

As mentioned at the beginning of this section, when the interpolation technique is used, it is sometimes sufficient for stability and convergence for less than  $2(R + s)_{\max}$  values and derivatives to be converted to the new stepsize. The requirements on the stepsize changing matrix  $C_m$  are that  $C_m = I + O(h)$  for stability, and that the stepsize operation has error  $O(h^C)$  for accuracy. This is achieved if the data used in the interpolating polynomial is chosen using  $r_m^0, r_m^1$  as above.

The formula changing technique is such that when we use a new formula which requires more past values (or derivatives) than the old one, we use actual past data rather than arbitrary values, such as zero. Formally, we are introducing a formula change matrix  $(O_{ij})$  in the notation of Gear and Watanabe (1974)), which is the identity. The result is equivalent to using a matrix which updates all the derivatives in the Nordsieck representation when the formula is changed.

We now prove the main theorem of this section.

Theorem 3.4 If a family of methods of the form (3.1) is used to solve the ODE (1.1) in a manner such that

1) each formula  $F_i$  is strongly stable,

2) each formula has order  $\geq 1$ ,

and 3) the interpolation technique with  $\tilde{c} = 2$  or the variable step technique is used;

then for each formula  $F_i$  there exists an integer  $p_i$  such that the method is stable and convergent for a stepsize selection scheme (2.12) that produces small stepsize changes satisfying

$$h_m/h_{m-1} = 1 + O(h)$$

and a formula selection scheme which produces infrequent formula changes in the sense that at least  $p_i$  steps are taken after the formula  $F_i$  has been selected.

Proof: The proof will use several technical lemmas shown by Gear and Tu (1974) and Gear and Watanabe (1974).

The key condition is the stability condition, i.e., that there exists  $k_0 < \infty$  such that

$$\hat{S}_n^m = \hat{S}_{m-1} \dots \hat{S}_n$$

satisfies

$$\|\hat{S}_n^m\| \leq k_0$$

for all  $0 \leq t_n < t_m \leq T$ .

Lemma 3.5 (Gear and Tu, 1974, Lemma 2.) If

$$S_n = \hat{S}_n + h_n \tilde{S}_n,$$

$\{\hat{S}_n\}$  satisfies the stability condition, and  $\tilde{S}_n = O(1)$  for all  $n$ , then there exists  $k_2$  such that  $\|S_n^m\| \leq k_2$  for all  $m \geq n$ .

The following lemma is proved in the same manner as Theorem 2 of Gear and Tu (1974):

Lemma 3.6 If a method satisfies the stability condition, if  $\tilde{S}_n$  (of Lemma 3.5) is uniformly bounded, and if the truncation error  $d_n$  satisfies

$$\|d_n\| \leq h_n e(h) \quad \text{for all } n$$

where  $e(h) \rightarrow 0$  as  $h \rightarrow 0$ , then the method converges and there exist constants  $k_2$  and  $k_3$  such that the global error  $\varepsilon_n$  satisfies

$$\|\varepsilon_n\| \leq k_2 \|\varepsilon_0\| + k_3 e(h) \quad \text{for } 0 \leq t_n \leq T.$$

Since each formula is of order  $\geq 1$ , the hypothesis on  $d_m$  is satisfied. Thus, showing the stability condition will prove convergence. Similarly by Theorem 1a of Gear and Tu (1974), the stability condition will imply stability.

Lemma 3.7 If a method satisfies the stability condition, and  $\tilde{S}_n$  (of Lemma 3.5) satisfies  $\|\tilde{S}_n\| \leq k_1 < \infty$  for all  $0 \leq t_n \leq T$ , then the method is stable.

The matrix  $\hat{S}_m$  depends only on the formula  $F_i$  used and is the amplification matrix for a constant stepsize method applied to  $y' = 0$  and may be denoted  $\dot{S}_i$ . Since the formulas are exact for  $y' = 0$ ,  $y(0) = 1$ ,  $\dot{S}_i$  has an eigenvector  $x_i$  corresponding to the eigenvalue 1 with ones corresponding to entries  $y_{n-j}$  in  $(Y_m, \dots, Y_{m-\rho_{\max}})^T$  or  $T_m$  (depending on the technique) and zeroes elsewhere. The formulas are strongly stable at  $h\lambda = 0$ , so other eigenvalues of  $\dot{S}_i$  are less than 1. The following lemma, which is a special case of Lemma 2.1 of Gear and Watanabe (1974), shows that  $\hat{S}_n$  can be bounded as long as there are enough  $\dot{S}_i$ 's of the same index side by side.

Lemma 3.8 Let  $\{\dot{S}_i\}$  be a set of matrices with the following properties:

- 1) Each  $\dot{S}_i$  has one simple eigenvalue equal to 1 with the same eigenvector  $x_1$ , and all other eigenvalues are less than 1 in magnitude.
- 2) If  $Q_i$  is such that its columns have norm 1, and  $Q_i^{-1} \dot{S}_i Q_i = J_i$ , the Jordan normal form of  $\dot{S}_i$  (with the first column of  $Q_i$  chosen to be  $x_1$ ), then there exist constants  $C_0, C_1$  such that

$$\|Q_i^{-1}\| \leq C_0, \quad \|Q_i\| \leq C_1 \quad \text{for all } i.$$

Then for any constant  $K > C_0 C_1$  there exists a set of integers  $\{P_i\}$  such that

$$\left\| \prod_{j=1}^N S_{ij}^{q_{ij}} \right\| \leq K \quad \text{for all } N \geq 1$$

whenever  $q_{ij} \geq P_{ij}$ .

This establishes Theorem 3.4.

Q.E.D.

We note that the hypotheses of the theorem are satisfied for all the methods in the examples of this paper. Thus for sufficiently infrequent formula changes and small stepsize changes, the methods are stable and convergent.

## 4. QUADRATURE METHODS

Methods which solve equation (1.6) in practice use a quadrature formula to estimate the integral. We choose the weights  $w_{ij}$  and abscissas  $\gamma_{ij}$  so that the quadrature formula,

$$p(t_n + \theta_i h_n) = p(t_{n-1}) + \sum_{j=1}^{\ell} w_{ij} f(p(\gamma_{ij}), \gamma_{ij}), \quad i = 0, \dots, s-1,$$

integrates polynomials with degree  $C - 1$  exactly.

To find the absolute stability region we apply the method to  $y' = \lambda y$ , with constant stepsize  $h$ . Then

$$\begin{aligned} p(t_n + \theta_i h) &= p(t_{n-1}) + \sum_{j=1}^{\ell} w_{ij} \lambda p(\gamma_{ij}) \\ &= p(t_{n-1}) + \int_{t_{n-1}}^{t_n + \theta_i h} \lambda p(\xi) d\xi. \end{aligned}$$

By (1.8), and changing the variable of integration to  $\tau = (\xi - t_n)/h$ ,

$$\sum_{jk} \phi_{jk}(\theta_i) \frac{h^k y_{n+j}^{(k)}}{k!} = y_{n-1} + \sum_{jk} \left[ h\lambda \int_{-1}^{\theta_i} \phi_{jk}(\tau) d\tau \right] \frac{h^k y_{n+j}^{(k)}}{k!}.$$

But  $y_{n+j}^{(k)} = \lambda^k y_{n+j}$ , so the preceding formula becomes

$$\begin{aligned} \sum_{jk} \phi_{jk}(\theta_i) \frac{(h\lambda)^k}{k!} y_{n+j} &= y_{n-1} + \sum_{jk} \frac{(h\lambda)^{k+1}}{k!} \int_{-1}^{\theta_i} \phi_{jk}(\tau) d\tau y_{n+j}, \quad (4.1) \\ i &= 0, \dots, s-1. \end{aligned}$$

For  $s > 1$ , this can be written as a vector difference equation, in which case the discussion of order, stability, and region of absolute stability presented in Chapter 3 applies.

The remainder of this chapter will be devoted to the special case of  $s = 1$ . Then (4.1) has the form

$$a_{-r}y_{n-r} + \dots + a_0y_n = 0 \quad (4.2)$$

where

$$a_j = \sum_{k=0}^{c_j-1} \left( \phi_{jk}(\theta_0) \frac{\mu^k}{k!} - \int_{-1}^{\theta_i} \phi_{jk}(\tau) d\tau \frac{\mu^{k+1}}{k!} \right) + \delta_{i,-1}.$$

Here, as elsewhere, we have  $\mu \equiv h\lambda$ . As before, the region of absolute stability is

$$\{\mu \mid \text{all roots } \xi \text{ of } \pi(\mu, \xi) = 0 \text{ have } |\xi| \leq 1 \text{ and } |\xi| = 1 \text{ implies } \xi \text{ is a simple root}\} \quad (4.3)$$

where

$$\pi(\mu, \xi) = a_{-r} + a_{-r+1}\xi + \dots + a_0\xi^r.$$

The specific methods whose stability regions are shown in this paper are all strongly stable at  $\mu = \infty$ , that is, the roots of  $\pi(\infty, \xi)$  are all less than 1 in magnitude. By continuity of the set of roots of a polynomial as a function of its coefficients, the methods are strongly stable in the exteriors of the locus

$$\{\mu \mid \text{there exists } \xi \text{ with } |\xi| = 1 \text{ such that } \pi(\mu, \xi) = 0\}.$$

To define the order of the method, let  $y_{n+i} = y(t_{n+i})$ ,  $i = -r, \dots, s-1$ , where  $y(t)$  is the exact solution of the ODE (1.1). Let  $p$  be defined as in (1.8) and let

$$F_n(y) = p(t_n + \theta_0 h_n) - p(t_{n-1}) - \sum_{j=1}^{\ell} w_{ij} f(p(\gamma_{ij}), \gamma_{ij})$$

be the amount by which the exact solution fails to satisfy the method.

Then the method is of order k if

$$D_n(y) = O(h^{k+1})$$

for all solutions  $y(t)$  which have at least  $k + 1$  continuous derivatives.

As was shown in the collocating case, this definition is equivalent to the general definition stated earlier in this chapter. By the properties of

interpolating polynomials and the Lipschitz property on  $f$ ,

$$p(t_n + \theta_0 h_n) = y(t_n + \theta_0 h_n) + O(h_n^C), \quad (4.4)$$

$$p(t_{n-1}) = y(t_{n-1}),$$

$$\begin{aligned} \int_{t_{n-1}}^{t_n + \theta_0 h_n} f(p(\xi), \xi) d\xi &= \int_{t_{n-1}}^{t_n + \theta_0 h_n} f(y(\xi), \xi) + O(h^C) d\xi \\ &= \int_{t_{n-1}}^{t_n + \theta_0 h_n} f(y(\xi), \xi) d\xi + O(h^{C+1}). \end{aligned}$$

Because the quadrature formula is assumed to be precision  $C - 1$ , the order of the method is at least  $C - 1$ . If  $\theta_0$  is such that  $t_n + \theta_0 h_n$  is an interpolation point, then  $p(t_n + \theta_0 h_n) = y(t_n + \theta_0 h_n)$ , so the order is at least  $C$ .

Two families of stiffly stable formulas are

$$I(0:1^r;2) \quad r = 1, \dots, 7 \quad (4.5)$$

and

$$I(0:2,1^{r-1};2) \quad r = 1, \dots, 6. \quad (4.6)$$

The stability regions of (4.5) are presented in Fig. 4.1. The corresponding  $D$ 's and  $\alpha$ 's are given in Table 4.1. The stability regions of (4.6) are presented in Fig. 4.2 with corresponding  $D$ 's and  $\alpha$ 's in Table 4.2. The formulas of (4.5) have the same stability polynomials as Enright's second derivative methods (1974) even though they do not require calculation of  $Df$ . These two families were discovered by Watanabe (personal communication).

The integral formulas

$$I(0:c_{-1};c_0) \quad (4.7)$$

when applied to  $y' = \lambda y$  give the difference equation

$$\pi_{-1}(\mu)y_{n-1} = \pi_0(\mu)y_n$$

where  $\deg(\pi_{-1}) = c_{-1}$  and  $\deg(\pi_0) = c_0$ . The order of the method is  $c_{-1} + c_0$ , so

$$y(t_n) = \frac{\pi_0(\mu)}{\pi_{-1}(\mu)} y(t_{n-1}) + O(h^{c_{-1}+c_0+1}).$$

The exact solution satisfies  $y(t_n) = e^\mu y(t_{n-1})$ , so that

$$e^\mu = \frac{\pi_0(\mu)}{\pi_{-1}(\mu)} + O(h^{c_{-1}+c_0+1}) \text{ as } h \rightarrow 0, \lambda \text{ fixed.}$$

Thus  $\pi_0(\mu)/\pi_{-1}(\mu)$  must be the  $[c_0, c_{-1}]$  Padé approximant to  $e^\mu$  (Wall, 1948, pp. 377, 378). Thus by results of Birkhoff and Varga (1965) and Ehle (1969), the methods  $I(0; c_{-1}; c_0)$  are A-stable if  $c_{-1} = c_0$  and strongly A-stable if  $c_0 = c_{-1} + 1$  or  $c_0 = c_{-1} + 2$ .

The formulas (4.7) are useful when the derivatives of  $f$  are easy to evaluate, for example when  $f$  is a time-independent linear function. In general, however, quadrature formulas are less attractive to implement than the collocation formulas in the multistep case, since for the latter one does not need to calculate quadrature weights and abscissas at each step.

Table 4.1

INTEG1  $I(0:1^r;2)$ 

r	order	D	$\alpha(^{\circ})$
1	3	0	90
2	4	0	90
3	5	-0.103	87.9
4	6	-0.526	82.0
5	7	-1.339	73.1
6	8	-2.728	60.0
7	9	-5.178	37.7

Table 4.2

INTEG2  $I(0:2,1^{r-1};2)$ 

r	order	D	$\alpha(^{\circ})$
1	4	0	90
2	5	-0.274	86.1
3	6	-0.888	79.1
4	7	-1.878	69.3
5	8	-3.460	55.0
6	9	-6.157	28.6

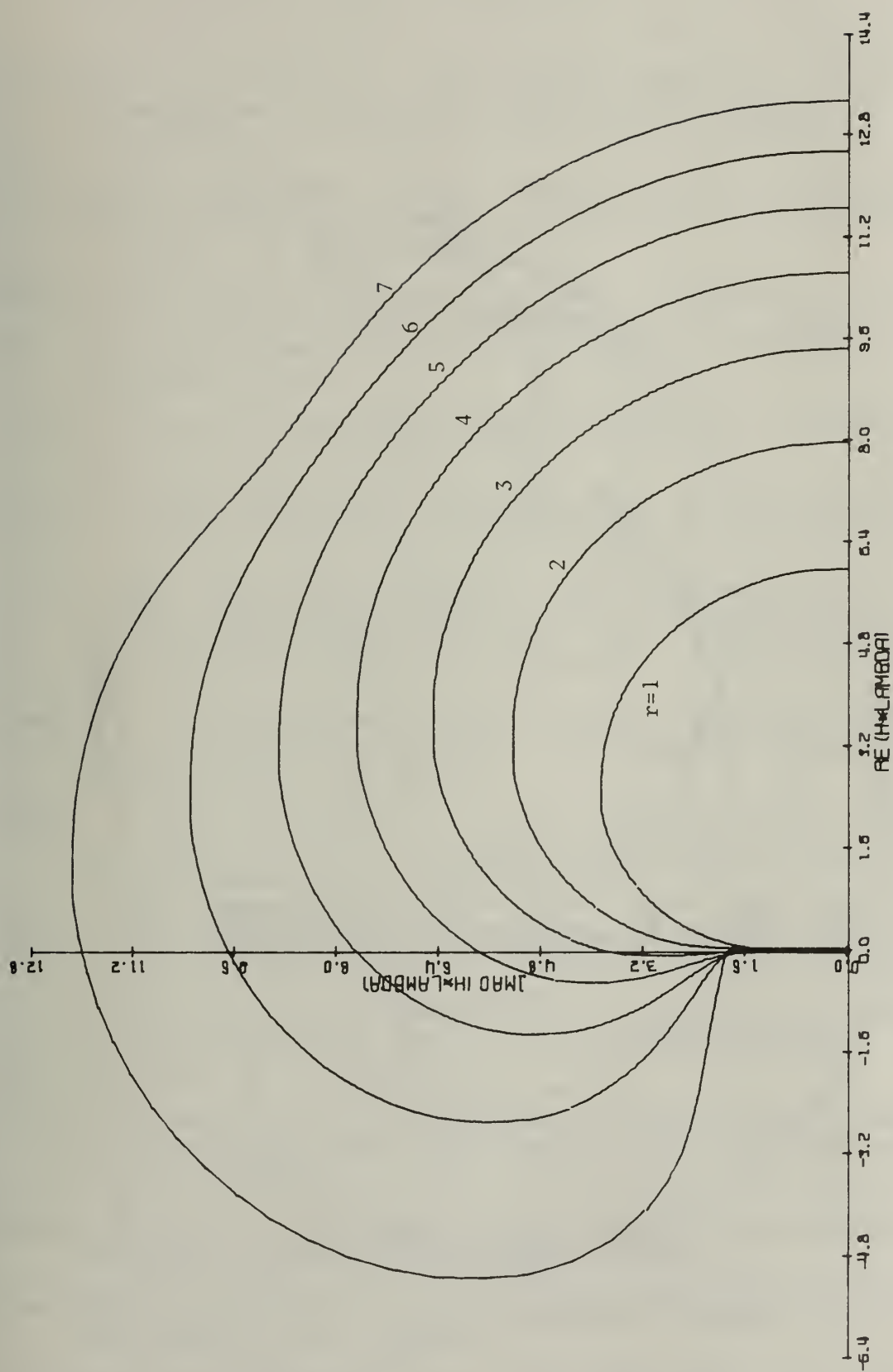


Fig. 4.1.1. INTEG1  $I(0; 1^r; 2)$ ,  $r = 1, \dots, 7$  Order 3-9

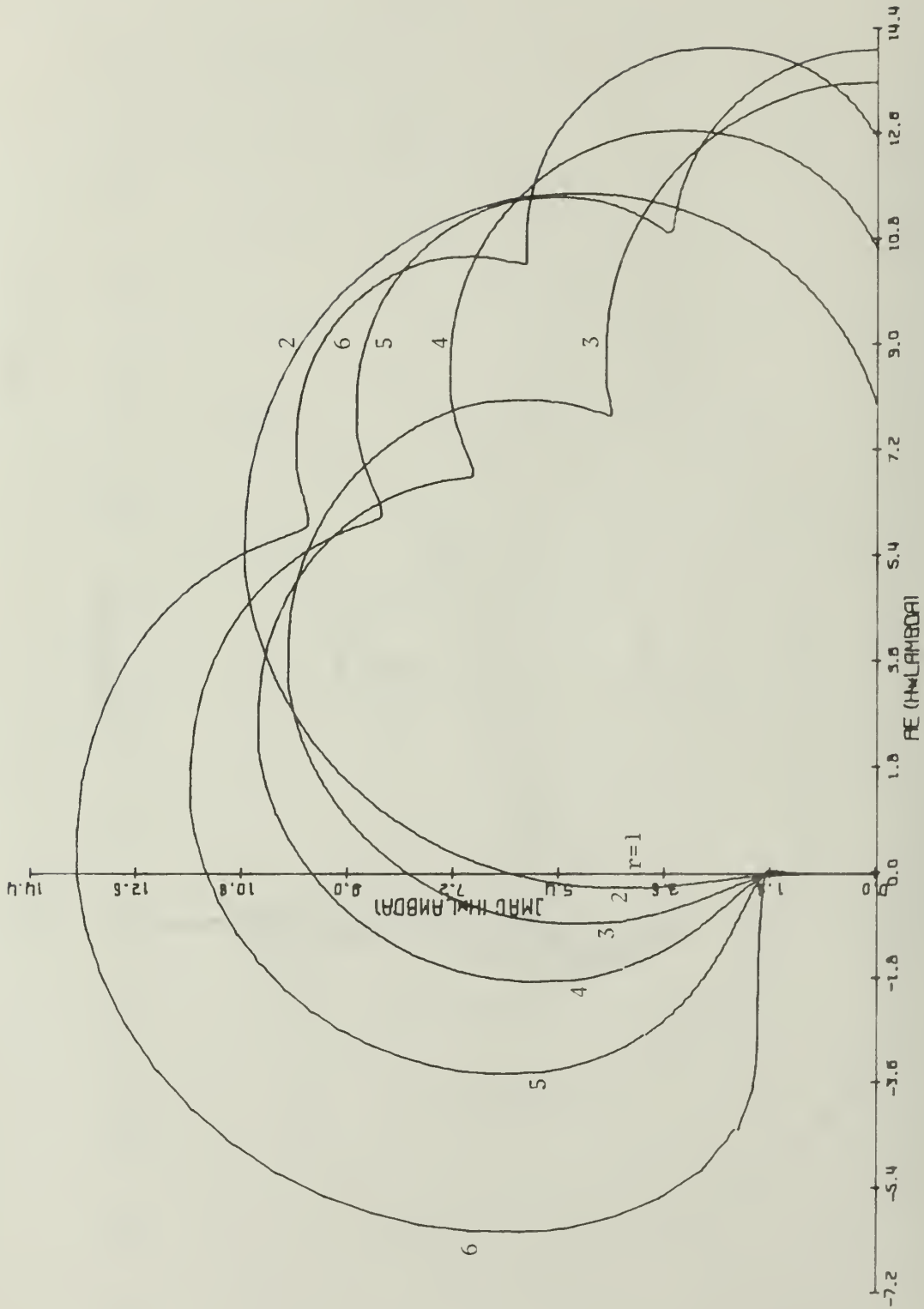


Fig. 4.2. INTEG2  $I(0;2,1^{r-1};2)$ ,  $r = 1, \dots, 6$  Order 4-9

## LIST OF REFERENCES

- Birkhoff, G., and Varga, R. S., 1965, "Discretization Errors for Well-Set Cauchy Problems, I," Journal of Mathematics and Physics, vol. 44, pp. 1-23.
- Bickart, T. A., and Picel, Z., 1973, "High Order Stiffly Stable Composite Multistep Methods for Numerical Intergration of Stiff Differential Equations," BIT, vol. 13, pp. 272-286.
- Bjurel, G.; Dahlquist, G.; Lindberg, B.; Linde, S.; and Oden, L., 1970, "Survey of Stiff Ordinary Differential Equations," Report # NA70.11, Dept. of Information Processing, Royal Inst. of Technology, Stockholm, Sweden.
- Cryer, C. W., 1973, "A New Class of Highly-Stable Methods:  $A_0$ -stable Methods," BIT, vol. 13, pp. 153-159.
- Ehle, B. L., 1968, "High Order A-stable Methods for the Numerical Solution of Systems of Differential Equations," BIT, vol. 8, pp. 276-278.
- , 1969, "On Pade Approximations to the Exponential Function and A-stable Methods for the Numerical Solution of Initial Value Problems," Research Report SCRR 2010, Dept. of Applied Analysis and Computer Science, University of Waterloo.
- Enright, W. H., 1974, "Second Derivative Multistep Methods for Stiff Ordinary Differential Equations," SIAM Journal on Numerical Analysis, vol. 11, pp. 321-331.
- Gear, C. W., 1969, "The Automatic Integration of Stiff Ordinary Equations," in Morrel, A. J. H., ed., Information Processing, 68, North Holland Pub. Co., Amsterdam, pp. 187-193.
- , 1971, Numerical Initial Value Problems in Ordinary Differential Equations, Prentice Hall, Inc., Englewood Cliffs, N. J.
- , and Tu, K. W., 1974, "The Effect of Variable Mesh Size on the Stability of Multistep Methods," SIAM Journal on Numerical Analysis, vol. 11, pp. 1025-1043.
- , and Watanabe, D. S., 1974, "Stability and Convergence of Variable Order Multistep Methods," SIAM Journal on Numerical Analysis, vol. 11, pp. 1044-1058.
- Henrici, P., 1962, Discrete Variable Methods in Ordinary Differential Equations, John Wiley and Sons, Inc., New York.
- Isaacson, E., and Keller, H. B., 1966, Analysis of Numerical Methods, John Wiley and Sons, Inc., New York.
- Jain, M. K., and Srivastava, V. K., 1970, "High Order Stiffly Stable Methods for Ordinary Differential Equations," Dept. of Computer Report No. 394, University of Illinois, Urbana, Illinois.

Jeltsch, R., 1976, "Stiff Stability and Its Relation to  $A_0$ - and  $A(0)$ -Stability," SIAM Journal on Numerical Analysis, vol. 13, pp. 8-17.

Wall, H. S., 1948, Analytic Theory of Continued Fractions, van Nostrand, New York.

## APPENDIX I

LISTING OF CODE FOR COLLOC1 FAMILY OF METHODS

```

COLLOC: PROC(N,T,Y,H,HMIN,HMAX,EPS,K,MAXDER,FAILFLAG,FAIL) REORDER;
  DCL (N,K,MAXDER,FAILFLAG) FIXED BIN,
      (T,Y(*,*),H,HMIN,HMAX,EPS) FLOAT(16),FAIL LABEL;
/*
  N          NO. OF FIRST ORCER DIFFERENTIAL EQUATIONS

  T          THE INDEPENDENT VARIABLE

  Y          CONTAINS THE DEPENDENT VARIABLES AND THEIR SCALED
             DERIVATIVES. Y(J,I) CONTAINS THE J-TH DERIVATIVE OF
             Y(I) MULTIPLIED BY H**J/FACTORIAL(J), WHERE H IS THE
             CURRENT STEPSIZE. ONLY Y(0,I) NEED BE PROVIDED BY THE
             CALLING PROGRAM ON INITIAL ENTRY.

  H          THE STEPSIZE TO BE ATTEMPTED ON THE NEXT STEP. H MAY BE
             ADJUSTED UP OR DOWN BY THE PROGRAM AFTER TAKING THE STEP

  HMIN       MINIMUM STEPSIZE THAT CAN BE USED

  HMAX       MAXIMUM STEPSIZE THAT CAN BE USED

  EPS        ERROR TEST CONSTANT. SINGLE STEP ERROR ESTIMATES MUST
             BE LESS THAN THIS IN NORM. THE STEPSIZE AND/OR ORDER
             IS ADJUSTED TO THIS END

  K          ORDER TO BE USED FOR ATTEMPTING NEXT STEP. IF K<1 AN
             INITIAL STEP WILL BE ATTEMPTED. OTHERWISE THE NEXT STEP
             WILL CONTINUE FROM THE LAST. UPON RETURN K HOLDS CURRENT
             ORDER AND INDEX OF HIGHEST AVAILABLE DERIVATIVE

  MAXDER     THE MAXIMUM DERIVATIVE THAT CAN BE USED, EQUAL TO THE
             HIGHEST ORDER. SHOULD BE <= 9

  FAILFLAG   A COMPLETION CODE WITH THE FOLLOWING MEANINGS:
             0 STEP WAS SUCCESSFUL

  FAIL       A LABEL TO WHICH CONTROL WILL BE TRANSFERED IN CASE
             A SUCCESSFUL STEP CANNOT BE TAKEN
*/
/* A PROCEDURE DIFFUN(T,H,Y,N) IS REQUIRED WHICH SETS
   Y(1,*) = H*F(T,Y(0,*))
   WHERE THE DIFFERENTIAL EON BEING SOLVED IS
   Y'=F(T,Y) */

  DCL (X(N),D,ALPHA,          SAVE(0:MAXDER,N),F(N))FLOAT(16),
      (ALPHAD,DD,ALPHAU,DU,ALPHAM) FLOAT(16),
      H_OLD STATIC INIT(0) FLOAT(16),
      ERRCOEF(2:9,3) FLOAT(16) STATIC INIT(
        *,          5E0,          1.48E0,
        2.50E0,      .971E0,      .444E0,
        2.96E0,      .392E0,      .219E0,
        6.76E0,      .233E0,      .147E0,
        22.7E0,      .161E0,      .110E0,
        99.9E0,      .121E0,      .0869E0,
        543.5E0,     .0955E0,     .0716E0,
        3516.E0,     .0783E0,     * ),
      INITIALIZE STATIC FIXED BIN INIT(0),
      /* INITIALIZE=0 TO REFRESH H,  =1 IF OLD H EXISTS */
      (I,J,STEP_CT STATIC INIT(0),FFLAG          )FIXED BIN;

```

```

IF K<2 THEN DO;
  CALL DIFFUN(T,H,Y,N);
  DO I=1 TO N; SAVE(0,I) = Y(0,I) + HMIN*Y(1,I)/H; END;
  CALL DIFFUN(T+HMIN,H,SAVE,N);
  DO I=1,N; Y(2,I) = (SAVE(1,I)-Y(1,I))*H/HMIN/2E0; END;
  K=2;
  H_OLD=0E0;
  INITIALIZE=0;
  END;
IF H /= H_OLD & H_OLD /= 0E0 THEN DO;
  ALPHA = H/H_OLD;
  CALL CHANGE_H;
  INITIALIZE = 0;
  END;

H_OLD = H;
DO I=1 TO N; DO J=0 TO K;
  SAVE(J,I) = Y(J,I);
END; END;
TRY: CALL DIFSTEP(N,T,Y,H,K);
D=0;
DO I=1 TO N;
  D = D+X(I)*X(I);
END;
D = SQRT(D)*ERRCOEF(K,2);
ALPHA = (EPS/D)**(1E0/(K+1))*0.9E0;

IF D > EPS THEN DO; /* STEP FAILED -- REPEAT */
  IF H=HMIN THEN GO TO F1;
  T = T-H;
  DO I=1 TO N; DO J=0 TO K;
    Y(J,I) = SAVE(J,I);
  END; END;
  H = MAX(H*ALPHA,HMIN);
  ALPHA = H/H_OLD;
  CALL CHANGE_H;
  GO TO TRY;
END;

ELSE IF STEP_CT<0 THEN DO; /*TOO SOON TO CHANGE K OR H */
  STEP_CT = STEP_CT+1;
  H_OLD = H;
  END;

ELSE DO; /*CALCULATE STEP & ORDER CHANGES */

  IF K>2 THEN DO; /* CALC ALPHA TO LOWER K */
    DD = 0E0;
    DO I=1 TO N;
      DD = DD + Y(K,I)*Y(K,I);
    END;
    DD = SQRT(DD)*ERRCOEF(K,1);
    ALPHAD = (EPS/DD)**(1E0/K)*0.9E0;
    END;
    ELSE ALPHAD=0E0;

  IF K<MAXDER THEN DO; /* CALC ALPHA TO INCREASE K */
    DU = 0E0;
    DO I=1 TO N;
      DU = DU + (Y(K+1,I)-X(I))**2;
    END;
    DU = SQRT(DU)*ERRCOEF(K,3);
    ALPHAU = (EPS/DU)**(1E0/(K+2))*0.9E0;
    END;
    ELSE ALPHAU = 0E0;

  ALPHAM = MAX(ALPHAD,ALPHA,ALPHAU);

```

```

      IF ALPHAM>1.1E0 THEN DO; /* CHANGE STEPSIZE &/| ORDER */
        IF ALPHAM=ALPHAD THEN K=K-1;
        ELSE IF ALPHAM=ALPHAU THEN DO;
          K=K+1;
          DO I=1 TO N;
            Y(K,I) = (Y(K-1,I)-(H/H_OLD)**(K-1)*
                      SAVE(K-1,I))/K;
          END;
        END;
        H_OLD = H;
        H = MIN(H*ALPHAM,HMAX);
        END;
      ELSE DO; /* CHANGE TOO SMALL TO BOTHER WITH */
        H_OLD = H;
        END;
      END;
    IF STEP_CT>=0 & K<MAXDER THEN DO I=1 TO N;
      Y(K+1,I) = X(I);
    END;
  RETURN;

```

```

CHANGE_H:  PROC; DCL (I,J) FIXED B(1,N), ALPHAP FLOAT(16);

```

```

  ALPHAP = 1E0;
  DO J=1 TO K;
    ALPHAP = ALPHAP*ALPHA;
    DO I=1 TO N;
      Y(J,I) = Y(J,I)*ALPHAP;
    END;
  END;
  STEP_CT = -K-3;
  END CHANGE_H;

```

```

F1:      FAILFLAG=11; /* ERROR TOO LARGE & H=HMIN */
        GO TO FAIL;

```

```

F2:      FAILFLAG = 2; /* NONLINA FAILED WITH H=HMIN */
        GOTO FAIL;

```

```

FA2:     IF H>HMIN THEN DO; /* INTERCEPT NONLINA FAILURE */
          T = T-H;
          DO I=1 TO N; DO J=0 TO K;
            Y(J,I) = SAVE(J,I);
          END; END;
          H = MAX(H/ 2E0,HMIN);
          ALPHA = H/H_OLD;
          CALL CHANGE_H;
          INITIALIZE = 0;
          GO TO TRY;
        END;
      ELSE GO TO F2;

```

```

DIFSTEP:PROC(N,T,Y,H,K)  REORDER;
      DCL (N,K) FIXED BIN.(T,H,Y(*,*))  FLOAT(16);

/*      AT ENTRY Y(J,I) CONTAINS AN APPROXIMATION TO THE J-TH
      DERIVATIVE OF Y(I) TIMES H**J/FACTORIAL(J), CORRESPONDING TO TIME
      T.  THE PROCEDURE TAKES A STEP OF SIZE H AT ORDER K AND RETURNS
      UPDATED VALUES OF T,Y.  IN ADDITION,X(1:N) IS RETURNED WITH
      THE DIFFERENCES BETWEEN CORRECTED AND PREDICTED VALUES OF Y
      AT NEW TIME T(= OLD T+H), I.E. X=YC-YP      */

      DCL (YTHETA(0:1,N), YTP(0:1,N), YPREDICTED(0:1,N),DELY(0:1,N))
      FLOAT(16),
      (I,J,L) FIXED BIN,
      TT(2:9,2:9,0:1) FLOAT(16) STATIC INIT(

/* ORDER 2 */
/*      -1                      1      */
      -1.0000000000000000E0,      1.0000000000000000E0,
      (14) 0E0.

/* ORDER 3 */
/*      -7/4                    3/2      */
/*      -3/4                    1/2      */
      -1.7500000000000000E0,      1.5000000000000000E0,
      -7.5000000000000000E-1,      5.0000000000000000E-1,
      (12) 0E0.

/* ORDER 4 */
/*      -85/36                  11/6      */
/*      -5/3                    1        */
/*      -11/36                  1/6      */
      -2.3611111111111111E0,      1.8333333333333333E0,
      -1.6666666666666667E0,      1.0000000000000000E0,
      -3.0555555555555556E-1,      1.6666666666666667E-1,
      (10) 0E0.

/* ORDER 5 */
/*      -415/144                25/12      */
/*      -755/288                35/24      */
/*      -119/144                5/12      */
/*      -25/288                1/24      */
      -2.8819444444444444E0,      2.0833333333333333E0,
      -2.6215277777777778E0,      1.4583333333333333E0,
      -8.263888888888889E-1,      4.166666666666667E-1,
      -8.680555555555556E-2,      4.166666666666667E-2,
      (8) 0E0.

/* ORDER 6 */
/*      -12019/3600             137/60      */
/*      -343/96                 15/8        */
/*      -2149/1440              17/24      */
/*      -133/480                1/8        */
/*      -137/7200              1/120      */
      -3.3386111111111111E0,      2.2833333333333333E0,
      -3.5729166666666667E0,      1.8750000000000000E0,
      -1.4923611111111111E0,      7.083333333333333E-1,
      -2.7708333333333333E-1,      1.2500000000000000E-1,
      -1.9027777777777778E-2,      8.333333333333333E-3,
      (6) 0E0.

```

/\* ORDER 7 \*/

/*	-13489/3600	49/20
	-16219/3600	203/90
	-6503/2880	49/48
	-1631/2880	35/144
	-1009/14400	7/240
	-49/14400	1/720 */
	-3.746944444444444E0.	2.450000000000000E0.
	-4.505277777777778E0.	2.255555555555556E0.
	-2.257986111111111E0.	1.020833333333333E0.
	-5.663194444444444E-1.	2.430555555555556E-1.
	-7.006944444444444E-2.	2.916666666666667E-2.
	-3.402777777777778E-3.	1.388888888888889E-3.
	(4) 0E0.	

/\* ORDER 8 \*/

/*	-726301/176400	363/140
	-9743/1800	469/180
	-311821/100800	967/720
	-17/18	7/18
	-8069/50400	23/360
	-179/12600	1/180
	-121/235200	1/5040 */
	-4.117352607709750E0.	2.592857142857143E0.
	-5.412777777777778E0.	2.605555555555556E0.
	-3.093462301587301E0.	1.343055555555556E0.
	-9.444444444444444E-1.	3.888888888888889E-1.
	-1.600992063492063E-1.	6.388888888888889E-2.
	-1.420634920634920E-2.	5.555555555555556E-3.
	-5.144557823129251E-4.	1.984126984126984E-4.
	(2) 0E0.	

/\* ORDER 9 \*/

/*	-3144919/705600	761/280
	-17763211/2822400	29531/10080
	-534731/134400	267/160
	-753029/537600	1069/1920
	-19637/67200	9/80
	-1379/38400	13/960
	-6779/2822400	1/1120
	-761/11289600	1/40320 */
	-4.457084750566893E0.	2.717857142857143E0.
	-6.293654691043084E0.	2.929662698412698E0.
	-3.978653273809524E0.	1.668750000000000E0.
	-1.400723586309524E0.	5.567708333333333E-1.
	-2.922172619047619E-1.	1.125000000000000E-1.
	-3.591145833333333E-2.	1.354166666666667E-2.
	-2.401856575963719E-3.	8.928571428571428E-4.
	-6.740717120181405E-5.	2.480158730158730E-5 );

/\* PREDICT \*/

DO L=0 TO K-1;

DO J=K-1 BY -1 TO L;

DO I=1 TO N; Y(J,I) = Y(J,I) + Y(J+1,I); END;

END;

END;

DO I=1 TO N; DO J=0,1;

YPREDICTED(J,I) = Y(J,I);

END; END;

```

/* FIND YTHETA CORRESPONDING TO YPREDICTED */

YTP(0,*) = YPREDICTED(0,*) + YPREDICTED(1,*) ;
YTP(1,*) = YPREDICTED(1,*) ;
DO J=2 TO K; DO I=1 TO N;
  YTP(0,I) = YTP(0,I) + Y(J,I);
  YTP(1,I) = YTP(1,I) + J*Y(J,I);
END; END;
T = T+H; X(*) = 0E0;
CALL NONLINA(0,N,EPS ,MAX(4*N,16),FA2,FFLAG,COMPUTE);
DO I=1 TO N;
  DO J=2 TO K;
    Y(J,I) = Y(J,I) + TT(K,J,0)*DELY(0,I)
              + TT(K,J,1)*DELY(1,I);
  END;
END;
RETURN;

```

```

COMPUTE: PROC(EQUS,FLOC) REORDER;
  DCL EQUS FIXED BIN,FLOC LABEL;
  DCL (I,J) FIXED BIN,T_THETA(2:9,0:1,0:1) FLOAT(16) STATIC INIT(

```

```

/* ORDER 2 */

```

/*	0	2
	-2	3 */
0.0000000000000000E0,		2.0000000000000000E0,
-2.0000000000000000E0,		3.0000000000000000E0,

```

/* ORDER 3 */

```

/*	-3/2	3
	-23/4	11/2 */
-1.5000000000000000E0,		3.0000000000000000E0,
-5.7500000000000000E0,		5.5000000000000000E0,

```

/* ORDER 4 */

```

/*	-10/3	4
	-197/18	25/3 */
-3.333333333333333E0,		4.000000000000000E0,
-1.094444444444444E1,		8.333333333333333E0,

```

/* ORDER 5 */

```

/*	-65/12	5
	-2501/144	137/12 */
-5.416666666666667E0,		5.000000000000000E0,
-1.736805555555556E1,		1.141666666666667E1,

```

/* ORDER 6 */
/*      -77/10      6
      -4973/200      147/10 */
      -7.700000000000000E0,      6.000000000000000E0,
      -2.486500000000000E1,      1.470000000000000E1,

/* ORDER 7 */
/*      -203/20      7
      -13327/400      363/20 */
      -1.015000000000000E1,      7.000000000000000E0,
      -3.331750000000000E1,      1.815000000000000E1,

/* ORDER 8 */
/*      -446/35      8
      -208903/4900      761/35 */
      -1.274285714285714E1,      8.000000000000000E0,
      -4.263326530612245E1,      2.174285714285714E1,

/* ORDER 9 */
/*      -4329/280      9
      -4134649/78400      7129/280 */
      -1.546071428571428E1,      9.000000000000000E0,
      -5.273786989795918E1,      2.546071428571428E1 );
ON OVERFLOW GOTO FLOC;

DO I=1 TO N;
  Y(0,I) = YPREDICTED(0,I) + X(I);
END;
CALL DIFFUN(T,H,Y,N);
DO I=1 TO N;
  DELY(0,I) = X(I);
  DELY(1,I) = Y(1,I)-YPREDICTED(1,I);
END;
DO I=1 TO N;
  YTHETA(0,I) = YTP(0,I) + T_THETA(K,0,0)*DELY(0,I)
               + T_THETA(K,0,1)*DELY(1,I);
END;
CALL DIFFUN(T+H,H,YTHETA,N);
DO I=1 TO N;
  F(I) =(YTP(1,I) + T_THETA(K,1,0)*DELY(0,I)
         + T_THETA(K,1,1)*DELY(1,I)
         - YTHETA(1,I) )/MAX(1E0,H);
END;
END COMPUTEF;
END DIFSTEP;

```

```

NONLINA:PROC(EQUUS,ORDER,TOL,MAXF,FAIL,TYPE,COMPUTE) REORDER;
/* BASIC ALGORITHM FROM COMP J VOL 12 NO 4 NOV 1969 PP 406-408 */
/* BY C.G.BROYDEN */

```

```

/* THIS PROCEDURE SOLVES THE NONLINEAR SIMULTANEOUS EQUATIONS
F(I)(X(1),X(2), ... ,X(N)) = 0, I=1,2, ... ,N.
IT ASSUMES THAT TWO N VECTORS, X AND F, AND A PROCEDURE
COMPUTE(EQUUS,FAIL) HAVE ALREADY BEEN DECLARED. USING THE CONTENTS
OF ARRAY X AS DATA THE PROCEDURE COMPUTE SHOULD CALCULATE THE
APPROPRIATE RESIDUALS AND ASSIGN THESE TO THE ARRAY F. BEFORE CALLING
NONLINA INITIAL VALUES OF THE ELEMENTS OF X SHOULD HAVE BEEN ASSIGNED.
OF THE TWO FORMAL PARAMETERS OF COMPUTE THE FIRST, AN INTEGER,
INDICATES WHICH SET OF NONLINEAR EQUATIONS IS TO BE SOLVED, I.E.
WHICH PARTICULAR MAPPING OF X ONTO F SHOULD BE CARRIED OUT BY COMPUTE
THIS ENABLES ANY ONE OF AN ARBITRARY NUMBER OF SETS OF NONLINEAR
EQUATIONS TO BE SOLVED. THE SECOND PARAMETER, FAIL, IS A LABEL TO
WHICH CONTROL SHOULD BE TRANSFERRED IN THE EVENT OF FAILURE DURING
EXECUTION OF COMPUTE. THE FORMAL PARAMETERS OF NONLINA ARE AS
FOLLOWS:

```

- (1) EQUUS(INTEGER), FULFILLS THE SAME FUNCTION AS THE FIRST FORMAL PARAMETER OF COMPUTE.
- (2) ORDER(INTEGER), THE NUMBER OF EQUATIONS AND UNKNOWN IN THE SET OF EQUATIONS SELECTED BY EQUUS.
- (3) TOL(FLOAT(16)), THE LARGEST ACCEPTABLE VALUE OF  $ABS(F)**2$ .
- (4) MAXF(FIXED BIN), THE MAXIMUM ACCEPTABLE NUMBER OF EVALUATIONS OF F.
- (5) FAIL(LABEL), THE LABEL OF THE FAILURE EXIT.
- (6) TYPE(FIXED BIN), SEE BELOW.

THE INTEGER TYPE IS A GUIDE TO THE KIND OF FAILURE THAT MAY HAVE OCCURRED. IT ASSUMES THE FOLLOWING VALUES:

0. NO FAILURE.
1. MAXIMUM NUMBER OF FUNCTION EVALUATIONS EXCEEDED. POSSIBLE CAUSES: TOL OR MAXF TOO SMALL, PROBLEM TOO NONLINEAR, INITIAL MATRIX TOO INACCURATE. POSSIBLE ACTION: INSPECT  $ABS(F)$  AND IF SMALL INCREASE EITHER MAXF OR TOL. IF  $ABS(F)$  LARGE USE NONLINB.
2. DIVISION BY ZERO WHILE UPDATING H, STORE ELEMENTS OF F IN DIFFERENT ORDER AND IF THIS FAILS USE NONLINB.
3. FAILURE IN COMPUTE. USE NONLINB.
4. DIVISION BY ZERO WHILE INITIALIZING H, COMPUTE ELEMENTS OF F IN DIFFERENT ORDER.
5. FAILURE IN COMPUTE WHILE INITIALISING H, CHOOSE IMPROVED INITIAL ESTIMATE OF SOLUTION.
6. OVERFLOW. IMPROVE INITIAL ESTIMATE \*/

```

DCL (EQUUS,ORDER,MAXF,TYPE) FIXED BIN;
DCL COMPUTE ENTRY(FIXED BIN,LABEL);
DCL TOL FLOAT(16), FAIL LABEL;
DCL (SA,SB,SC) FLOAT(16);
DCL (I,J,K,FCOUNT) FIXED BIN;
DCL (OLD_ORDER) STATIC INIT(0) FIXED BIN;
/* OLD_ORDER HOLDS DIMENSIONS OF CURRENT H */
DCL ((Z,P,V)(ORDER),H(ORDER,ORDER) CTL) FLOAT(16);
ON OVERFLOW GOTO F6;

```

```

STEP:      PROC(F1,F2);
           DCL (F1,F2) LABEL;
           DO I=1 TO ORDER;
             X(I) = X(I) + P(I);
             V(I) = F(I);
           END;
           CALL COMPUTEF(EQUS,F1); FCOUNT = FCOUNT+1;
           DO I=1 TO ORDER; Z(I) = F(I)-V(I); END;
           SA=0;
           DO I=1 TO ORDER;
             SB=0;
             DO J=1 TO ORDER;
               SB = SB + H(I,J)*Z(J);
             END;
             V(I) = SB-P(I); SA = SA+SB*P(I);
           END /*CALCULATION OF HZ-P AND PHY */ ;
           IF SA=0 THEN DO;
             DO I=1 TO ORDER;
               SA = SA+F(I)*F(I);
             END;
             IF SA<TOL THEN GOTO EXIT;
             DO I=1 TO ORDER;
               IF ABS(V(I)) > 1E-70 THEN GO TO F2;
             END;
             RETURN;          /* NO MODIFICATION OF H */
             END;
           DO J=1 TO ORDER;
             SB = 0;
             DO I=1 TO ORDER;
               SB = SB + P(I)*H(I,J);
             END;
             SB = SB/SA;
             DO I=1 TO ORDER;
               H(I,J) = H(I,J)-SB*V(I);
             END; /*MODIFICATION OF H */
           END;
           END STEP;

           IF ORDER /= OLD_ORDER THEN DO;
             OLD_ORDER = ORDER;
             FREE H;
             ALLOCATE H;
             INITIALIZE =0;
             END;

```

```

RESTART:   TYPE = 0;
           CALL COMPUTEF(EQUS,F5);
           IF INITIALIZE = 0 THEN DO;

           /* INITIALIZE ITERATION MATRIX H */
           DO I=1 TO ORDER;
             P(I) = 0; H(I,I) = 1E0;
             DO J=I+1 TO ORDER; H(I,J),H(J,I) = 0; END;
           END;
           DO K=1 TO ORDER;
             P(K) = EPS*MAX(EPS,ABS(Y(0,K)));CALL STEP(F5,F4);
             P(K) = 0E0;
           END;

           END;
           FCOUNT = 1;
           DO I=1 TO ORDER;
             SA = 0;
             DO J=1 TO ORDER; SA = SA-H(I,J)*F(J); END;
             P(I) = SA;
           END /* CALCULATION OF STEP VECTOR P */;

```

```

REPEAT:      CALL STEP(F3,F2);
              SB = 0;  SX=0;
              DO I=1 TO ORDER;
                  SA = 0;
                  DO J=1 TO ORDER; SA = SA-H(I,J)*F(J); END;
                  P(I) = SA;
                  SB = MAX(SB,ABS(SA));
                  SX = MAX(SX,ABS(X(I)));
              END;
              SA=0;
              DO I=1 TO ORDER; SA = SA + F(I)*F(I); END;
              IF DFLAG>=10 & T>40E0 THEN
                  IF SA <1E-10& SB/SX < .01E0 THEN GOTO EXIT;
                  IF FCOUNT >= MAXF THEN GOTO F1;
              GOTO REPEAT;
F6:          TYPE = 6;  GOTO FAIL;
F5:          TYPE=5; GOTO FAIL;
F4:          TYPE=4; GOTO FAIL;
F3:          TYPE=3; GOTO FAIL;
F2:          TYPE=2; GOTO FAIL;
F1:          TYPE=1;
              IF INITIALIZE /= 0 THEN DO;
                  INITIALIZE=0;
                  X(*) = 0E0;
                  GO TO RESTART;
              END;
              ELSE GO TO FAIL;

EXIT:        IF FCOUNT < MAXF/2 THEN INITIALIZE=1; ELSE INITIALIZE=0;
              END NONLINA;
              END COLLOC;

```

## APPENDIX II

## LISTING OF CODE TO FIND AND PLOT STABILITY REGIONS

```
INPUT: PROC(RR,SS,RS,METHOD,C,C_MAX,THETA,R_MAX,S_MAX);
```

```
/* FORMAT OF INPUT:
```

```
'S=NN,R=NN; (THETA0, ..., THETA(S-1):C(-R), ..., C(-1);
C(0), ..., C(S-1))'
*/
```

```
DCL (RR,SS,RS,C(*)) FIXED BIN,METHOD CHAR(*);
DCL(R_MAX,S_MAX) FIXED BIN;
DCL C_MAX FIXED BIN,THETA(*) FLOAT(16),DESCR CHAR(400)VARYING;
DCL STRING CHAR(400) VARYING, CHAR CHAR(1),
    TEMPLATE CHAR(400) VARYING,
    BUFFER CHAR(80) VARYING,
I FIXED DECIMAL,(J,K,L,M,NUM,R,S) FIXED BIN;
```

```
READ: GET LIST(DESCR);
PUT PAGE EDIT(DESCR)(A);
METHOD = DESCR;
GET STRING(DESCR) DATA(R,S);
STRING=''; /*IN CASE OF ERROR IN Q,R,S */
IF R<0 | S<1 | R>R_MAX | S>S_MAX
    THEN CALL ERROR(5);
RR=R; SS=S; /*CANT GET DATA WITH PARAMETERS */
RS = CEIL(R/S);
J=INDEX(DESCR,'(');
IF J=0 THEN CALL ERROR(1);
STRING=SUBSTR(DESCR,J)||'$';

/* $ TO INSURE STRING NONEMPTY */

/* GET THETA(I) */

DO I=0 TO S-1;
    IF SUBSTR(STRING,1,1) = ':' THEN CALL ERROR(2);
    STRING = SUBSTR(STRING,2);
    K = VERIFY(STRING,'0123456789.E+-');
    IF K<2 THEN CALL ERROR(3);
    BUFFER = SUBSTR(STRING,1,K-1);
    STRING = SUBSTR(STRING,K);
    CHAR = SUBSTR(STRING,1,1);
    IF CHAR ^= '.' & CHAR ^= ':' THEN CALL ERROR(4);
    GET STRING(BUFFER||',') LIST(THETA(I));
END;
IF CHAR ^= ':' THEN CALL ERROR(6);
```

```
/* GET C(J) */
```

```
STRING = SUBSTR(STRING,2); /* CHECK SYNTAX */
TEMPLATE=TRANSLATE(STRING,'DDDDDDDDDD$','0123456789D');
CHAR = SUBSTR(TEMPLATE,1,1);
M=0; /*NO ; SEEN */
L=0; /*NUMBER OF C(I)S */
NUM=0; /*1=SEEN NUM; -1=SEEN COMMA OR SEMI*/
```

```

NEXT_NUM:      DO WHILE(CHAR=' '); CALL BUMP; END;
               IF CHAR=';' THEN DO;
                   CALL BUMP;
                   M=M+1;
                   IF M>1 THEN CALL ERROR(8);
                   IF L≠R THEN CALL ERROR(7);
                   IF NUM<0 THEN CALL ERROR(11);
                   NUM=-1;
                   END;
               ELSE IF CHAR='D' THEN DO;
                   IF NUM>0 THEN CALL ERROR(4);
                   NUM=1;
                   L=L+1;
                   DO WHILE(CHAR='D'); CALL BUMP; END;
               ELSE IF CHAR='.' THEN DO;
                   IF NUM<1 THEN CALL ERROR(11);
                   NUM=-1;
                   CALL BUMP;
                   END;
               ELSE IF CHAR=')' THEN DO;
                   IF M=0 THEN CALL ERROR(8);
                   IF L≠R+S THEN CALL ERROR(9);
                   IF R=0 THEN STRING=TRANSLATE(STRING,'.,,;))');
                           ELSE STRING=TRANSLATE(STRING,'.,, . ;))' );
                   GET STRING(STRING) LIST((C(J) DO J=-R TO S-1));
                   C_MAX = C(-R);
                   DO I=-R+1 TO S-1; C_MAX = MAX(C_MAX,C(I)); END;
                   RETURN;
                   END;
               ELSE CALL ERROR(4);
               GOTO NEXT_NUM;
BUMP:  PROC;
      TEMPLATE=SUBSTR(TEMPLATE,2);
      CHAR=SUBSTR(TEMPLATE,1,1);
      END;

ERROR: PROC(MESS_NUMBER); DCL MESS_NUMBER FIXED BIN;
      DCL MESSAGE(12) STATIC CHAR(30) VARYING INIT(
          'NO (.,,
          'TOO FEW THETAS.,
          'VACUOUS THETA.,
          'IMPROPER DELIMITER.,
          'BAD VALUE FOR Q, R, OR S.,
          'TOO MANY THETAS.,
          'WRONG NUMBER OF PAST C(I)S.,
          'WRONG NUMBER OF ;',
          'WRONG NUMBER OF FUTURE C(I)S.,
          *,
          'VACUOUS C(I)S.,
          'C_MAX TOO LARGE');

      PUT SKIP(2) EDIT(MESSAGE(MESS_NUMBER), ' INPUT IGNORED.')(A);
      PUT SKIP(2) EDIT(STRING)(A);
      GOTO READ;
      END;
      END INPUT;

```

```

STABPOL: PROC(R,S,RS,C,C_MAX,THETA,P,MU_MAX,XI_MAX,NEXT_METHOD);
  DCL

  THETA(*) FLOAT(16), /*OFF-STEP COLLOCATION POINTS */
  (R,          /* NUMBER OF PAST POINTS */
  S,          /* NUMBER OF POINTS IN ADVANCED BLOCK */
  C(*),       /* METHOD INTERPOLATES TO (C(I)-1)ST
              DERIVATIVE AT X(N+I) */
  RS,        /* NUMBER OF PAST BLOCKS*/
  C_MAX,     /*MAXIMUM C(I) */
  MU_MAX,    /*DEGREE OF STABILITY POLY IN MU  */
  XI_MAX     /*DEGREE OF STABILITY POLY IN XI */
  ) FIXED BIN,
  /*PERTINENT DIMENSIONS OF C ARE (-R:S-1) */
  P(*,*) FLOAT(16), /*HOLDS COFS OF STABILITY POLY*/

  /* STABILITY PLOYNOMIAL IS SUM P(I,J)*(MU**I)*(XI**J)
     I=0, ..., MU_MAX; J=0, ..., XI_MAX */
  NEXT_METHOD LABEL;
  DCL N FIXED BIN; /* DIMENSION OF INTERPOLATING POLYS*/

  /* ELEMENT(I,J) IS COEF OF (XI**I)(MU**J) */
  DCL 1 A(0:S-1,0:S-1),
      2 EL(0:C_MAX,0:RS) FLOAT(16),
      POLY(0:S*C_MAX,0:S*RS) FLOAT(16);
  A = 0E0; POLY = 0E0;
  CALL SET_UP_A(R,S,C,RS,N);
  CALL FIND_DET_A(R,S,RS,C_MAX,P,MU_MAX,XI_MAX);
  RETURN;

```

```

SET_UP_A: PROC(R,S,C,RS,N);
  DCL (R,S,C(*),RS,N) FIXED BIN;
  DCL (I,J,L,K,JJ,INDEX,X(0:50)) FIXED BIN;
  DCL (TAB(0:50),V,D) FLOAT(16);
  DCL FAC(0:10) FLOAT(16) INIT(1E0,1E0,2E0,6E0,24E0,120E0,
    720E0,5040E0,40320E0,362880E0,3628800E0);
  IF C_MAX>10 THEN DO;
    PUT SKIP EDIT('C_MAX TOO LARGE.')(A);
    GOTO NEXT_METHOD;
  END;

  N=-I; /*FIND DEGREE OF INTERPOLATING POLY */
  DO I=-R TO S-I;
    N=N+C(I);
  END;
  IF N>50 THEN DO;
    PUT SKIP EDIT('DEGRÉE OF INTERPOLATING ',
      'POLYNOMIAL TOO LARGE')(2 A);
    GOTO NEXT_METHOD;
  END;

```

```

INDEX=0;                      /* SET UP ORDINATES FOR DIVIDED
                               DIFFERENCE TABLE */
DO I=-R TO S-1;
  DO J=1 TO C(I);
    X(INDEX) = I;
    INDEX=INDEX+1;
  END;
END;
DO J=-R TO S-1;
  K=RS+FLOOR(J/S);
  JJ=MOD(J,S);
  DO L=0 TO C(J)-1;
    CALL TABLE(J,L,N,X);
    DO I=0 TO S-1;
      CALL DERIVS(THETA(I),V,D);
      /* LET(A(I,J) = A(I,J)+P(1)/FAC(L)*MU**L
        -P(0)/FAC(L)*MU**(L+1) */
      A(I,JJ).EL(L,K) =
        A(I,JJ).EL(L,K) +D/FAC(L);
      A(I,JJ).EL(L+1,K) =
        A(I,JJ).EL(L+1,K) - V/FAC(L);
    END;
  END;
END;
RETURN;

TABLE:      PROC(II,JJ,N,X);
DCL (I,J,II,JJ,N,X(*)) FIXED BIN,(CUR,LAST) FLOAT(16);

/* COMPUTES TABLE OF DIVIDED DIFFERENCES TO CHARACTERIZE
   PHI(II,JJ). AT RETURN, TAB(J),J=0,...,N
   IS A FORMAC ARRAY WHICH CONTAINS THE LOWER
   RISING DIAGONAL OF THE TABLE WITH THE
   ORDINATE X(0) AT THE BOTTOM */

DO I=N BY -1 TO 0;
  DO J=0 TO N-I;
    IF X(I)=X(I+J) & J=JJ & X(I)=II
      THEN CUR=1;
    ELSE IF X(I)=X(I+J)
      THEN CUR=0;
    ELSE CUR=(LAST-TAB(J-1))/
      (X(I+J)-X(I));
    IF J<N-I THEN LAST=TAB(J);
    TAB(J) = CUR;
  END;
END;
END TABLE;

```

```

DERIVS:      PROC(XX,V,D);
              DCL J FIXED BIN,XX FLOAT(16),
                  (W(0:N),V,D,LAST,CUR) FLOAT(16);

              /*USING CURRENT INSTANCE OF TAB COMPUTE ASSOCIATED
                POLYNOMIAL AND 1 DERIVATIVE EVALUATED AT XX */
              /* V IS VALUE AND D IS DERIVATIVE */

              W(0)=1;V=TAB(0);
              DO J=1 TO N;
                  W(J) = W(J-1)*(XX-X(J-1));
                  V = V + TAB(J)*W(J);
              END;

                  LAST = W(1);
                  W(1) = W(0);
                  D = W(1)*TAB(1);
                  DO J=2 TO N;
                      CUR = W(J-1)*(XX-X(J-1))+LAST;
                      LAST = W(J);
                      W(J) = CUR;
                      D = D+W(J)*TAB(J) ;
                  END;
              END DERIVS;
END SET_UP_A;

```

```

FIND_DET_A: PROC(R,S,RS,C_MAX,P,MU_MAX,XI_MAX);
    DCL (R,S,RS,C_MAX) FIXED BIN,P(*,*) FLOAT(16),
        (MU_MAX,XI_MAX) FIXED BIN;
    DCL(I,J,K,J_MAX) FIXED BIN;

    /* FORM POLY=DET(A) */

    /*
DO I=0 TO S-1;
DO J=0 TO S-1;
    PUT SKIP(2) EDIT('A(',1,J,') = ')(A,2 F(1),A);
    PUT SKIP;
    DO L=0 TO C_MAX;
        PUT SKIP;
        DO K=0 TO RS;
            PUT EDIT(A(I,J),EL(L,K))(E(20,12));
        END;
    END;
END;END; */

```

```

IF S=1 THEN DO I=0 TO C_MAX;
    DO J=0 TO RS;
        POLY(I,J) = A(0,0).EL(I,J);
    END;END;
ELSE IF S=2 THEN CALL CROSS(0,0,1,POLY);
ELSE IF S=3 THEN BEGIN;
    DCL (COF0,COF1,COF2)(0:2*C_MAX,0:2*RS) FLOAT(16);
    CALL CROSS(1,1,2,COF0);
    CALL CROSS(1,0,2,COF1);
    CALL CROSS(1,0,1,COF2);
    CALL SUM(POLY,0,0,COF0);
    CALL DIF(POLY,1,0,COF1);
    CALL SUM(POLY,2,0,COF2);
    END;
ELSE IF S=4 THEN BEGIN;
    DCL (C1,C2,C3,C4,C5,C6)(0:2*C_MAX,0:2*RS) FLOAT(16);
    (COF0,COF1,COF2,COF3)(0:3*C_MAX,0:3*RS) FLOAT(16);
    CALL CROSS(2,0,1,C1);
    CALL CROSS(2,0,2,C2);
    CALL CROSS(2,0,3,C3);
    CALL CROSS(2,1,2,C4);
    CALL CROSS(2,1,3,C5);
    CALL CROSS(2,2,3,C6);
    COF0=0E0; COF1=0E0; COF2=0E0; COF3=0E0;

    CALL SUM(COF0,1,1,C6);
    CALL DIF(COF0,2,1,C5);
    CALL SUM(COF0,3,1,C4);

    CALL DIF(COF1,0,1,C6);
    CALL SUM(COF1,2,1,C3);
    CALL DIF(COF1,3,1,C2);

    CALL SUM(COF2,0,1,C5);
    CALL DIF(COF2,1,1,C3);
    CALL SUM(COF2,3,1,C1);
    CALL DIF(COF3,0,1,C4);
    CALL SUM(COF3,1,1,C2);
    CALL DIF(COF3,2,1,C1);

    CALL SUM(POLY,0,0,COF0);
    CALL SUM(POLY,1,0,COF1);
    CALL SUM(POLY,2,0,COF2);
    CALL SUM(POLY,3,0,COF3);
    END;
ELSE DO;
    PUT SKIP(2) EDIT('S>4 NOT IMPLIMENTED')
        (A);
    GO TO NEXT_METHOD;
    END;
XI_MAX = S*RS;
MU_MAX = S*C_MAX;
DO I=0 TO MU_MAX;
    DO J=0 TO XI_MAX;
        P(I,J) = POLY(I,J);
    END;
END;
PUT SKIP(2) EDIT('MU_MAX = ',MU_MAX,', XI_MAX = ',XI_MAX)
    (A,F(2),A,F(2));
PUT SKIP(2);
DO K=0 BY 5 TO XI_MAX;
    J_MAX = MIN(K+4,XI_MAX);
    PUT SKIP(3) EDIT(('XI**',J DO J=K TO J_MAX))(X(6),(J_MAX-K+1)
        (X(8),A,F(2),X(8)));

```

```

PUT SKIP(2) EDIT(('MU**',I,(P(I,J) DO J=K TO J_MAX) DO I=0 TO
MU_MAX)) (A,F(2),(J_MAX-K+1)E(22.14),SKIP);
END;

```

```

CROSS: PROC(KK,II,JJ,RESULT);

```

```

/* FORM DETERMINANT
   RESULT = A(II, KK)*A(JJ, KK+1)-A(JJ, KK)*A(II, KK+1)
*/
DCL RESULT(*,*) FLOAT(16), (II, JJ, KK) FIXED BIN;
DCL (I, J, K, L, M1, M2, KKP1, I2, J2) FIXED BIN;
KKP1=KK+1;
DO L=0 TO 2*RS;
M1=MAX(0, L-RS); M2 = MIN(L, RS);
DO K=0 TO 2*C_MAX;
  RESULT(K, L)=0E0;
  DO I=MAX(0, K-C_MAX) TO MIN(K, C_MAX);
    I2 = K-I;
    DO J=M1 TO M2;
      J2 = L-J;
      RESULT(K, L) = RESULT(K, L)
        + A(II, KK).EL(I, J)*
          A(JJ, KKP1).EL(I2, J2)
        - A(JJ, KK).EL(I, J)*
          A(II, KKP1).EL(I2, J2);
    END;
  END;
END;
END CROSS;

```

```

SUM: PROC(AA, II, JJ, C);

```

```

DCL (AA, C) (*, *) FLOAT(16);

```

```

/* FORM A = A+B*C AS POLYS IN 2 VARIABLES */

```

```

DCL (I, J, K, L, M1, M2, II, JJ, ADD, I2, J2, B1, B2, C1, C2) FIXED BIN;
DCL TEMP FLOAT(16);
ADD=1;

```

```

COMMON: B1=C_MAX; B2=RS;

```

```

C1=HBOUND(C, 1); C2=HBOUND(C, 2);

```

```

IF HBOUND(AA, 1)<B1+C1 | HBOUND(AA, 2)<B2+C2 THEN DO;
  PUT SKIP EDIT('A TOO SMALL', B1, B2, C1, C2) (A, 4 F(10));
  STOP;
END;

```

```

DO L=0 TO C2+B2;

```

```

PUT SKIP;

```

```

M1=MAX(0, L-C2); M2=MIN(L, B2);

```

```

DO K=0 TO C1+B1;

```

```

  TEMP=0E0;

```

```

  DO I=MAX(0, K-C1) TO MIN(K, B1);

```

```

    I2=K-I;

```

```

    DO J=M1 TO M2;

```

```

      J2=L-J;

```

```

      TEMP = TEMP + A(II, JJ).EL(I, J)*C(I2, J2);

```

```

    END;
  END;

```

```

IF ADD=1 THEN AA(K, L)=AA(K, L)+TEMP;

```

```

ELSE AA(K, L) = AA(K, L)-TEMP;

```

```

END;
END;

```

```

RETURN;

```

```

DIF: ENTRY(AA, II, JJ, C);

```

```

ADD=0;

```

```

GOTO COMMON;

```

```

END SUM;

```

```

END FIND_DET_A;

```

```

END STABPOL;

```

```

FINDRTS:      PROC(P,MU_MAX,XI_MAX,X,Y,NUM_POINTS,D,PSI,NEXT_METHOD);
              DCL P(*,*) FLOAT(16), /*COFS OF STABILITY POLY*/
              (MU_MAX, /*DEGREE OF POLY IN MU */
              XI_MAX, /*DEGREE OF POLY IN XI */
              NUM_POINTS)FIXED BIN,(D,PSI) FLOAT,

              (X(*,*),Y(*,*)) FLOAT, /*HOLDS STABILITY LOCUS*/
              NEXT_METHOD LABEL;
              CALL INPUT;
              CALL XI_AT_INF(P,MU_MAX,XI_MAX);
              CALL FIND_ROOTS(P,MU_MAX,XI_MAX,X,Y,NUM_POINTS,D,PSI);
              RETURN;

INPUT: PROC;
          DCL (I,J,I0,J0) FIXED BIN;

          /* REMOVE TRIVIAL ZEROES */
TEST_XI:DO J0=0 TO XI_MAX;
          DO I=0 TO MU_MAX;
              IF P(I,J0)~=0EO THEN GOTO TEST_MU;
          END;
        END;
TEST_MU:DO I0=0 TO MU_MAX;
          DO J=J0 TO XI_MAX;
              IF P(I0,J) ~= 0EO THEN GOTO REDUCE;
          END;
        END;
REDUCE:IF I0=0 & J0=0 THEN RETURN;
          DO I=I0 TO MU_MAX;
              DO J=J0 TO XI_MAX;
                  P(I-I0,J-J0) = P(I,J);
              END;
          END;
          MU_MAX = MU_MAX-I0;
          XI_MAX = XI_MAX-J0;
          IF XI_MAX=0 THEN DO;
              PUT SKIP EDIT('XI_MAX = 0. METHOD SKIPPED')(A);
              GO TO NEXT_METHOD;
          END;
        END INPUT;

```

```

XI_AT_INF:PROC(P,MU_MAX,XI_MAX);
  DCL P(*,*) FLOAT(16),(MU_MAX,XI_MAX) FIXED BIN;
  DCL TRAUBZ ENTRY(FIXED BIN(31),(*)COMPLEX FLOAT(16),(*)COMPLEX
    FLOAT(16),(*)FLOAT(16),(*)COMPLEX FLOAT(16),(*)COMPLEX
    FLOAT(16),(*)COMPLEX FLOAT(16),(*)COMPLEX FLOAT(16),
    (*)COMPLEX FLOAT(16),FIXED BIN(31)) OPTIONS(FORTRAN);
  DCL (C(21),ZCAL(20),DUM2(20),DUM3(20),DUM4(19),DUM5(19),
    DUM6(19)) COMPLEX FLOAT(16),
    DUM1(20) FLOAT(16),(N,NRTFND) FIXED BIN(31);
  DCL (I,NUM_ZERO) FIXED BIN;

  /* XI'S AT INFINITY ARE THE ROOTS OF THE COEFFICIENT OF
    MU**MU_MAX, WHICH IS A POLYNOMIAL IN XI */

  DO N=XI_MAX BY -1 TO 0 WHILE(P(MU_MAX,N)=0E0); END;
  IF N<1 THEN GOTO STABLE;
  DO I=0 TO N;
    C(I+1) = P(MU_MAX,N-I);
  END;
  DO I=N BY -1 TO 0 WHILE(C(I+1)=0E0);END;
  NUM_ZERO=N-I;
  N=I;
  IF N<1 THEN GOTO STABLE;
  CALL TRAUBZ(N,C,ZCAL,DUM1,DUM2,DUM3,DUM4,DUM5,DUM6,NRTFND);
  IF NRTFND<N THEN PUT SKIP(2) EDIT(N-NRTFND,' ROOTS NOT FOUND')
    (F(2),A);
  PUT SKIP(2) EDIT('XI FOR MU=INFINITY:')(A);
  DO I=N+1 TO N+NUM_ZERO;
    ZCAL(I)=0E0;
  END;
  N=N+NUM_ZERO;
  PUT SKIP(2) EDIT((ZCAL(I),'I,' DO I=1 TO NRTFND))
    (3 (C(E(20,11),E(18,11)),A),SKIP);
  DO I=1 TO NRTFND;
    IF ABS(ZCAL(I))>1.000001E0 THEN GOTO UNSTABLE;
  END;
  DO I=1 TO NRTFND;
    IF ABS(ZCAL(I))>0.999999E0 THEN GOTO MARG_STABLE;
  END;
  STABLE:PUT SKIP(2) EDIT('STABLE AT INFINITY')(A);
  RETURN;
  MARG_STABLE:PUT SKIP(2) EDIT('Marginally stable at infinity')(A);
  GO TO NEXT_METHOD;
  UNSTABLE:PUT SKIP(2) EDIT('UNSTABLE AT INFINITY. METHOD SKIPPED.')(A);
  GO TO NEXT_METHOD;
  END XI_AT_INF;

```

```

FIND_ROOTS:PROC(P,MU_MAX,XI_MAX,X,Y,NUM_ROOTS,D,PSI);
  DCL P(*,*) FLOAT(16),(MU_MAX,XI_MAX) FIXED BIN,
      (X(*,*),Y(*,*)) FLOAT,NUM_ROOTS FIXED BIN;
  DCL TRAUBZ ENTRY(FIXED BIN(31),(*)COMPLEX FLOAT(16),(*)COMPLEX
      FLOAT(16),(*)FLOAT(16),(*)COMPLEX FLOAT(16),(*)COMPLEX
      FLOAT(16),(*)COMPLEX FLOAT(16),(*)COMPLEX FLOAT(16),
      (Z,XI) COMPLEX FLOAT(6),FIXED BIN(31)) OPTIONS(FORTRAN);
  DCL (C(21),ZCAL(20),DUM2(20),DUM3(20),DUM4(19),DUM5(19),
      DUM6(19)) COMPLEX FLOAT(16),
      DUM1(20) FLOAT(16),(N,NRTFND) FIXED BIN(31);
  DCL (THETA,DEL_THETA,DIST,DTEMP) FLOAT(16),
      (D,PSI,PTEMP,PI INIT(3.14159))FLOAT(6),
      (TOTAL,ITERATIONS,CALLS) FIXED BIN INIT(0),
      (Z,XI) COMPLEX FLOAT(6),(I,J,INDEX,K,L) FIXED BIN;

  /*HAVE ELIMINATED marginally stable at infinity cases so we
    know infinity is not a value of MU for |XI|=1.  thus
    we know
        SUM P(MU_MAX,J)*(XI**J) = 0
    FOR ALL |X|=1      */

  /* FIND ROOTS AT XI=1 */
  DO I=0 TO MU_MAX;
    C(I+1)=0;
    DO J=0 TO XI_MAX;
      C(I+1) = C(I+1)+P(MU_MAX-I,J);
    END;
  END;
  N=MU_MAX;
  CALL TRAUBZ(N,C,ZCAL,DUM1,DUM2,DUM3,DUM4,DUM5,DUM6,NRTFND);
  IF NRTFND<N THEN PUT SKIP(2) EDIT(N-NRTFND,' ROOTS NOT FOUND',
      ' FOR XI=1')(F(2),A,A);
  DO I=1 TO NRTFND;
    COMPLEX(X(0,I),Y(0,I)) = ZCAL(I);
  END;
  DO I=NRTFND+1 TO N;
    X(0,I),Y(0,I)=0E0;
  END;

  /* FIND ROOTS AT XI=EXP(DEL_THETA*I) */

  THETA,DEL_THETA = 3.141592653589793E0/NUM_ROOTS;
  CALL SET_UP_C(COMPLEX(COS(THETA),SIN(THETA)));
  CALL TRAUBZ(N,C,ZCAL,DUM1,DUM2,DUM3,DUM4,DUM5,DUM6,NRTFND);
  IF NRTFND<N THEN DO;PUT SKIP(2) EDIT(N-NRTFND,' ROOTS NOT FOUND',
      ' FOR XI=EXP(1E-5)')(F(2),A,A);
    GO TO NEXT_METHOD;
  END;

  /* ORDER ROOTS SO THEY ARE CLOSE TO PREVIOUS ONES */

  DO I=1 TO N;
    DIST = 1E70;
    Z = COMPLEX(X(0,I),Y(0,I));
    DO J=1 TO N;
      DTEMP = ABS(Z-ZCAL(J));
      IF DTEMP<DIST THEN DO;
        INDEX = J;
        DIST = DTEMP;
      END;
    END;
    COMPLEX(X(1,I),Y(1,I)) = ZCAL(INDEX);
    ZCAL(INDEX) = 1E70;
  END;
END;

```

```
/* SET UP FOR GENERAL CASE */
```

```
THETA = THETA+DEL_THETA;
CALL SET_UP_C(COMPLEX(COS(THETA),SIN(THETA)));
DO I=1 TO N;
    Z = COMPLEX(2*X(1,I)-X(0,I),2*Y(1,I)-Y(0,I));
    COMPLEX(X(2,I),Y(2,I)) = NEWTON(Z);
END;
```

```
/* GENERAL CASE */
```

```
DO K=3 TO NUM_ROOTS;
    THETA = THETA+DEL_THETA;
    CALL SET_UP_C(COMPLEX(COS(THETA),SIN(THETA)));
    DO I=1 TO N;
        Z = COMPLEX(3*X(K-1,I)-3*X(K-2,I)+X(K-3,I),
                    3*Y(K-1,I)-3*Y(K-2,I)+Y(K-3,I));
        COMPLEX(X(K,I),Y(K,I)) = NEWTON(Z);
    END;
END;
```

```
/* FIND D, THETA */
```

```
D=1E70; PSI=0E0;
DO I=0 TO NUM_ROOTS;
    DO J=1 TO N;
        IF X(I,J)<D THEN D=X(I,J);
        IF X(I,J)**2+Y(I,J)**2 > 1E-12 THEN DO;
            PTEMP=ATAN(ABS(Y(I,J)),X(I,J));
            IF PTEMP>PSI THEN PSI=PTEMP;
        END;
    END;
END;
PSI = (3.1415926536-PSI)*180E0/3.1415926536;
PUT SKIP(2) EDIT('D = ',D,' THETA = ',PSI,' AVERAGE NEWTON',
    ' ITERATIONS = ',TOTAL/CALLS)
    (A,F(14,8),X(2),A,F(10,6),X(5),2 A,F(9,6));
```

```
/* PRINT OUT ROOTS */
```

```
DO K=1 BY 4 TO N;
    L=MIN(N,K+3);
    PUT SKIP(3) EDIT(('ROOT',I DO I=K TO L))
        (X(11),A,F(2),X(13));
    PUT SKIP;
    DO I=0 TO NUM_ROOTS;
        PUT SKIP EDIT((X(I,J),Y(I,J) DO J=K TO L))
            (2 F(14,8),X(2));
    END;
END;
RETURN;
```

```

SET_UP_C:PROC(XI);
  DCL XI COMPLEX FLOAT(6);
  DCL TEMP COMPLEX FLOAT(6),(I,J) FIXED BIN;
/* EVALUATES THE COEFFS OF THE STABILITY POLYNOMIAL AS A
POLY IN MU BY PLUGGING IN XI. NOTE ORDER OF COEFFS IS REVERSED
IN GOING FROM P TO C */

  DO I=0 TO MU_MAX;
    TEMP = P(I,XI_MAX);
    DO J=XI_MAX-1 BY -1 TO 0;
      TEMP = TEMP*XI+P(I,J);
    END;
    C(MU_MAX-I+1) = TEMP;
  END;
END SET_UP_C;

NEWTON:PROC(Z) RETURNS(COMPLEX FLOAT(6));
  DCL Z COMPLEX FLOAT(6);
  DCL (B(21),DERIV,Z0,Z1) COMPLEX FLOAT(16),
      I FIXED BIN ;
  CALLS=CALLS+1;
  Z1=Z;
  ITERATIONS=0;
LOOP:  ITERATIONS=ITERATIONS+1;
  Z0=Z1;
  B(1) = C(1);
  DO I=2 TO N+1;
    B(I) = B(I-1)*Z0+C(I);
  END;
  DERIV= B(1);
  DO I=2 TO N;
    DERIV = DERIV*Z0+B(I);
  END;
  Z1 = Z0-B(N+1)/DERIV;
  IF ABS(Z1-Z0)/MAX(ABS(Z1),ABS(Z0)) > 1E-6 & ITERATIONS<20
    THEN GOTO LOOP;
  IF ITERATIONS=20 THEN DO;
    PUT SKIP(3) EDIT('NEWTON FAILED')(A);
    GOTO NEXT_METHOD;
  END;
  TOTAL=TOTAL+ITERATIONS;
  RETURN(Z1);
END NEWTON;
END FIND_ROOTS;
END FINDRTS;

```

```

PLOTPTS:PROC(X,Y,NUM_POINTS,NUM_ROOTS,METHOD);
  DCL (X(*,*),Y(*,*)) FLOAT,(NUM_POINTS,NUM_ROOTS) FIXED BIN;
  DCL METHOD CHAR(*);
  DCL(CCP1PL ENTRY(FLOAT,FLOAT,FIXED BIN(31)),
      CCP4SC ENTRY((*)FLOAT,FLOAT,FIXED BIN(31),FIXED BIN(31),
          (2) FLOAT),
      CCP5AX ENTRY(FLOAT,FLOAT,CHAR(*),FIXED BIN(31),FLOAT,
          FLOAT,(2)FLOAT),
      SYMBOL ENTRY(FLOAT,FLOAT,FLOAT,CHAR(*),FLOAT,FIXED BIN(31))
      )OPTIONS(FORTRAN);
  DCL (SCX(2),SCY(2),EXT(2),TEMP,X_MAX,X_MIN,X_DIST)FLOAT,
      (Y_MAX,Y_DIST) FLOAT,
      (I,J) FIXED BIN,(M1,M2) CHAR(46);

  /* DETERMINE SCALE SO Y AXIS IS 8 INCHES */
  Y_MAX=0E0;X_MIN=1E70;X_MAX=-1E70;
  DO I=0 TO NUM_POINTS;
    DO J=1 TO NUM_ROOTS;
      Y_MAX = MAX(Y_MAX,ABS(Y(I,J)));
      X_MAX = MAX(X_MAX,X(I,J));
      X_MIN = MIN(X_MIN,X(I,J));
    END;
  END;
  EXT(1) = 0E0;
  EXT(2) = Y_MAX;
  CALL CCP4SC(EXT,8E0,2,1,SCY);
  SCX(2) = SCY(2);
  SCX(1) = FLOOR(X_MIN/SCX(2))*SCX(2);
  X_DIST = CEIL((X_MAX-SCX(1))/SCX(2));
  Y_DIST=8E0;
  IF X_DIST>13E0 THEN DO;
    EXT(1) = X_MIN;
    EXT(2) = X_MAX;
    CALL CCP4SC(EXT,13E0,2,1,SCX);
    SCY(2) = SCX(2);
    SCY(1) = 0E0;
    X_DIST = 13E0;
    Y_DIST = CEIL(Y_MAX/SCY(2));
  END;
  CALL CCP5AX(0E0,0E0,'RE(H*LAMBDA)',-12,X_DIST,0E0,SCX);
  CALL CCP5AX(-SCX(1)/SCX(2),0E0,'IMAG(H*LAMBDA)',14,
      Y_DIST,90E0,SCY);
  I=INDEX(METHOD,' ');
  M1 = SUBSTR(METHOD,I,46);
  M2 = SUBSTR(METHOD,I+46,46);
  CALL SYMBOL(0E0,-.75E0,.25E0,M1,0E0,46);
  CALL SYMBOL(0E0,-1.125,.25E0,M2,0E0,46);
  DO J=1 TO NUM_ROOTS;
    CALL CCP1PL((X(0,J)-SCX(1))/SCX(2),ABS(Y(0,J))/SCY(2),3);
    DO I=1 TO NUM_POINTS;
      CALL CCP1PL((X(I,J)-SCX(1))/SCX(2),
          ABS(Y(I,J))/SCY(2),2);
    END;
  END;
  END;
  CALL CCP1PL(X_DIST+2E0,0E0,-3);
  RETURN;
END PLOTPTS;

```

## V I T A

Bruce David Link was born July 9, 1948, in Seattle, Washington. He graduated from Penncrest High School, Media, Pennsylvania, in 1966. Mr. Link was elected to Phi Beta Kappa while attending the University of Oregon as a National Merit Scholar. During this time he worked as a Senior Systems Programmer at Oregon Research Institute in Eugene, Oregon. He graduated from the University of Oregon in 1970, receiving the degree of Bachelor of Arts (Honors College). Mr. Link was a National Science Foundation Trainee at the University of Illinois from September of 1970 until May of 1974, and served as a research assistant in the Department of Computer Science from August of 1973 until July of 1976. He also received the degrees of Bachelor of Science in Electrical Engineering, with highest honors, and Master of Science in Mathematics from the University of Illinois.

U. S. ATOMIC ENERGY COMMISSION  
UNIVERSITY-TYPE CONTRACTOR'S RECOMMENDATION FOR  
DISPOSITION OF SCIENTIFIC AND TECHNICAL DOCUMENT

( See Instructions on Reverse Side )

1. AEC REPORT NO.  COO-2383-0032	2. TITLE  NUMERICAL SOLUTION OF STIFF ORDINARY DIFFERENTIAL EQUATIONS USING COLLOCATION METHODS
--	--

3. TYPE OF DOCUMENT (Check one):

- ☒ a. Scientific and technical report
- ☐ b. Conference paper not to be published in a journal:  
Title of conference \_\_\_\_\_  
Date of conference \_\_\_\_\_  
Exact location of conference \_\_\_\_\_  
Sponsoring organization \_\_\_\_\_
- ☐ c. Other (Specify) \_\_\_\_\_

4. RECOMMENDED ANNOUNCEMENT AND DISTRIBUTION (Check one):

- ☒ a. AEC's normal announcement and distribution procedures may be followed.
- ☐ b. Make available only within AEC and to AEC contractors and other U.S. Government agencies and their contractors.
- ☐ c. Make no announcement or distribution.

5. REASON FOR RECOMMENDED RESTRICTIONS:

6. SUBMITTED BY: NAME AND POSITION (Please print or type)

C. W. Gear  
Professor and Principal Investigator

Organization

University of Illinois at Urbana-Champaign  
Department of Computer Science  
Urbana, IL 61891

Signature 	Date  June 1976
--	-----------------------

FOR AEC USE ONLY

7. AEC CONTRACT ADMINISTRATOR'S COMMENTS, IF ANY, ON ABOVE ANNOUNCEMENT AND DISTRIBUTION RECOMMENDATION:

8. PATENT CLEARANCE:

- ☐ a. AEC patent clearance has been granted by responsible AEC patent group.
- ☐ b. Report has been sent to responsible AEC patent group for clearance.
- ☐ c. Patent clearance not required.



BIBLIOGRAPHIC DATA SHEET		1. Report No. UIUCDCS-R-76-813	2.	3. Recipient's Accession No.	
4. Title and Subtitle  NUMERICAL SOLUTION OF STIFF ORDINARY DIFFERENTIAL EQUATIONS USING COLLOCATION METHODS				5. Report Date June 1976	
				6.	
7. Author(s) Bruce David Link				8. Performing Organization Rept. No. UIUCDCS-R-76-813	
9. Performing Organization Name and Address  Department of Computer Science University of Illinois at Urbana-Champaign Urbana, IL 61801				10. Project/Task/Work Unit No. COO-2383-0032	
				11. Contract/Grant No.  US ERDA E(11-1) 2383	
2. Sponsoring Organization Name and Address  Energy Research and Development Administration 9800 South Cass Avenue Argonne, IL 60439				13. Type of Report & Period Covered	
				14.	
5. Supplementary Notes					
6. Abstracts  A new class of methods, collocation methods, is suitable for the integration of stiff ordinary differential equations. The order and stability regions of these methods are characterized. The methods are stable and convergent for infrequent formula changes and small stepsize changes. Block multistep methods which are stiffly stable up to order 24 and several families of multistep methods stiffly stable up to order 9 exist. A similar class of quadrature methods includes a family with the same stability regions as Enright's second derivative methods but only requires first derivatives.					
7. Key Words and Document Analysis. 17a. Descriptors  Ordinary differential equations Collocation methods Block methods Hermite interpolation Stiffly stable					
7b. Identifiers/Open-Ended Terms					
7c. COSATI Field/Group					
8. Availability Statement  Unlimited				19. Security Class (This Report) UNCLASSIFIED	
				21. No. of Pages 113	
				20. Security Class (This Page) UNCLASSIFIED	
				22. Price	

















UNIVERSITY OF ILLINOIS-URBANA



3 0112 039572828